Terminais

1	write
2	while
3	variavel
4	until
5	to
6	then
7	string
8	repeat
9	real
10	read
11	program
12	procedure
13	or
14	of
15	nomeprocedure
16	literal
17	integer
18	if
19	î
20	for
21	end
22	else
23	do
24	declaravariaveis
25	const
26	char
27	chamaprocedure
28	begin
29	array
30	and
31	>=
32	>
33	=
34	<>
35	<=
36	<
37	+
38	_numreal
39	_numinteiro
40	_nomevariavel
41	_nomestring
42	_nomeprograma
43	_nomeprocedure

44 _nomechar
45]
46 [
47 ;
48 :
49 /
50 ..
51 .
52 ,
53 *
54)
55 (
56 $
57 -

Não Terminais

58 PROGRAMA
59 BLOCO
60 DCLPROC
61 DCLCONST
62 DCLVAR
63 CORPO
64 TIPO
65 LDCONST
66 LID
67 LDVAR
68 REPIDENT
69 TIPOARRAY
70 DEFPAR
71 COMANDO
72 REPCOMANDO
73 EXPRESSAO
74 ELSEPARTE
75 VARIAVEL
76 REPVARIAVEL
77 PARAMETROS
78 ITEMSAIDA
79 REPITEM
80 TERMO
81 REPEXP
82 REPEXPSIMP
83 FATOR
84 REPTERMO
85 EXPSIMP

MANUAL DA LINGUAGEM

Estrutura do Programa

Todo programa deve começar com a declaração *program* seguida do nome do programa, terminando a declaração com ‘; ’. O programa termina com o sinal de ponto final ‘.’.

Sintaxe:
program NOME;
##COMANDOS
.

Exemplo:
program soma;
##COMANDOS
.

Comentários

Os comentários são utilizados para explicar o código. Há dois tipos de comentários, os comentários de linha e de bloco (comentários que utilizam mais de uma linha).

De linha:
Isto é um comentário de linha.

De bloco:
```
#* Isto é um
    comentário
    de bloco.
*#
```

Tipos de dados

São aceitos os seguintes tipos de dados: integer para inteiros (1,2,3), char para caracteres (a,b,c), string para cadeia de caracteres (soma, resultado), real para números reais (1.2, 3.4, 5.7). Também existe a opção de declarar arrays, que funcionam como várias variáveis separadas, mas são declaradas como uma variável, e são controlados através de uma posição ([3..4]).

Declaração de Constantes e Variáveis

Define nomes de variáveis e os tipos de dados que a variável recebe. Pode-se declarar apenas uma variável por tipo, ou várias do mesmo tipo, na mesma linha. A declaração de variáveis começa com a tag declaravariaveis.

Uma variável por tipo:

Sintaxe:
declaravariaveis
NOME: TIPO;

Exemplo:
declaravariaveis
num: integer;
letra: char;

Declaração de array:
Sintaxe:
declaravariaveis
NOME: array [NUMEROLINHAS..NUMEROCOLUNAS] of TIPOARRAY;
Exemplo:
declaravariaveis
num: array[5..5] of integer;

Várias variáveis por tipo:
Sintaxe:
declaravariaveis
NOME, NOME, NOME: TIPO;

Exemplo:
declaravariaveis
num, n, resul: integer;
a, b, c: array[2..2] of char;

Constantes

Constantes são variáveis com um valor fixo.

Sintaxe:
const VARIAVEL = TIPO;

Exemplo:
const num = integer;

Operadores Aritméticos

Esta linguagem aceita os seguintes operadores aritméticos:

Adição: +
Subtração: -
Multiplicação: *
Divisão: /

Exemplo:
Declaravariaveis
num: real;
write(real*10)
Operadores de comparação e atribuição

Operadores de comparação comparam termos. Os operadores de comparação são:
Igual: =.
Diferente: <>.
Maior que: >.
Menor que: <.
Maior ou igual que: >=.
Menor ou igual que: <=.

Exemplos:
a = b
Verdadeiro (TRUE) se a é igual a b.
a <> b
Verdadeiro se a não é igual a b.
a > b
Verdadeiro se a é maior que b.
a < b
Verdadeiro se a é menor que b.
a >= b
Verdadeiro se a é maior ou igual a b.
a <= b
Verdadeiro se a é menor ou igual a b.

Operadores lógicos

São operadores que verificam se as expressões são verdadeiras ou falsas. Os operadores são: or e and.

Sintaxe:
a or b: Verdadeiro se a ou b são verdadeiros.
a and b: Verdadeiro se a e b são verdadeiros.

Entrada de dados

O comando responsável por reconhecer a entrada de dados é o read. A informação vai ser armazenada na variável entre parênteses.

Sintaxe:
read(VARIAVEL);

Exemplo:
read(num);

Pode-se ler mais de uma variável no comando read.

Sintaxe:
read(VARIAVEL, VARIAVEL);

Exemplo:
read(num, nume);

Saída de dados

Esse é o comando responsável por mostrar na tela as informações para o usuário. Podem ser impressos os dados de uma variável ou uma mensagem.

ITEMSAIDA: pode conter um texto (e se for deve estar entre aspas duplas), conteúdo de uma variável ou uma expressão aritmética.

Sintaxe:
write(ITEMSAIDA);

Exemplo:
write(“Hello World”,x+32);

Estruturas de controle

If/else

Os comandos if/else são as estruturas condicionais utilizadas nesta linguagem. Elas são utilizadas para verificar se uma determinada expressão é verdadeira, se for o bloco de código entre os comandos begin e end é executado. Se existir o comando else e a expressão no if for falsa, ele será executado.

Sintaxe:
if [EXPRESSAO] then
begin
##COMANDO
end

```
else
begin
##COMANDO
end;
```

Exemplo:

```
if [a<10] then
begin
        write(soma)
end
else
begin
        write(subtracao)
end;
```

while/do

O comando while é uma estrutura de repetição onde existe uma condição e enquanto esta condição for verdadeira, será executado os comandos entre begin e end. A condição deve estar entre colchetes.

Sintaxe:

```
while[EXPRESSAO] do
begin
##COMANDO
end;
```

Exemplo:

```
while [a<10] do
begin
        write(a)
end;
```

repeat / until

Repeat é um comando de repetição semelhante ao while, porém no repeat a condição vem depois do comando, e os comandos serão executados pelo menos uma vez. Assim os comandos serão executados até que a expressão no until seja falsa.

Sintaxe:

```
repeat
##COMANDO
until [EXPRESSAO];
```

Exemplo:
```
repeat
        write(soma)
until [soma<10];
```

for / to / do

É um comando de repetição que possui um ponto de partida e um ponto final conhecidos. A ponto final é testado a cada repetição, e enquanto for verdadeira o código é executado. O incremento ou decremento é feito entre os comandos begin e end.

Sintaxe:
```
for [variavel=EXPRESSAO] to [EXPRESSAO] do
begin
##CODIGO
end;
```

Exemplo:
```
for [cont = 1] to [10] do
begin
        write(cont)
end;
```

Procedures

As procedures tem a função de trazer uma parte do código que não se encontra na parte principal, ou seja, faz uma ligação das partes do código. A declaração de uma procedure com apenas um parâmetro deve ser feita da seguinte forma:

Sintaxe:
```
procedure NOME (VARIAVEL: TIPO)
declaravariaveis
NOME: TIPO;
begin
        ##comandos
end
```

Exemplo:
```
procedure mostra (num: integer)
declaravariaveis
a: integer;
begin
        write(num+a);
end
```

Se houver mais de um parâmetro deve-se primeiro mostrar os do mesmo tipo separadas por vírgula, seguidas de dois pontos: e do tipo, na seqüência um ponto e vírgula e o nome do próximo parâmetro e seu tipo, assim:

Sintaxe:
```
procedure soma (n, nn: integer; x: char)
declaravariaveis
r: integer;
begin
        write(n+nn);
end
```

Para se chamar uma procedure utiliza-se os seguintes comandos:

Sintaxe:
```
chamaprocedure NOME(VARIAVEL, VARIAVEL);
```

Exemplo:
```
chamaprocedure soma(a,b);
```

Regras Léxicas

- Os dados do tipo integer podem conter de informações inteiras de 0 à 500000;
- Os dados do tipo real podem conter de informações de 0 à 500000, onde se existir parte decimal, vai conter sempre duas casas após o ponto. Para separar a parte decimal da inteira deve-se utilizar o caractere ponto ‘.’;
- Os dados do tipo char podem receber apenas um caractere, se for mais de um caractere terá que ser usado o tipo string;
- Os comentários de uma linha deverão ser antecedidos por # #, e os de bloco, ou seja, que tem mais de uma linha devem ser iniciados de # * e terminados por * #;
- Os nomes de variáveis devem conter apenas caracteres, com no máximo 10 caracteres.
- O literal deve estar entre aspas duplas “”.

Erro léxico

Será considerado erro léxico:

- Se a variável contiver mais de 10 caracteres ou números;
- Se variável do tipo real estiver usando algum outro caractere que não for ponto ‘.’ Para separar a parte inteira da decimal;

- Se variável do tipo real ou integer conter números negativos ou maiores que 500000;
- Se variável do tipo char conter mais que um caracter;
- Se o usuário abrir um comentário de bloco e não fechar;
- Se o usuário abrir um literal e não fechar;
- Identificadores não podem ser palavras reservadas da linguagem.