**Definições de projeto**

- O programa deve conter inicialmente a palavra **program,** caso não tenha nada mais será analisado.

- É necessário seguir a ordem dos blocos: bloco de variáveis, depois constantes e por ultimo funções (podem existir um ou vários blocos de funções).

* ***Bloco Var:***

Deve estar logo abaixo da palavra reservada **“program”**, é inicializado com a palavra **var** deve possuir **begin** para inicializar e **end** para finalizar.

No bloco de variáveis não são aceitas expressões, nem identificadores nas inicializações. Aceita: “string” ‘a’ 1 [“matriz”]

É possível declarar múltiplas variáveis do mesmo tipo, desde que estejam separadas por vírgula.

* ***Bloco Const:***

-Constantes devem ser atribuídas.

-No bloco de constantes não são aceitas expressões, nem identificadores nas inicializações.

- É possível declarar múltiplas constantes do mesmo tipo, desde que estejam separadas por vírgula.

* ***Matriz:***

- Declaração de matrizes devem seguir o formato : id[numero]

ex: matriz[1][2].

- Atribuição de matriz deve seguir o formato: [<parâmetro>,< parâmetro>]

ex: [“string”] ou [‘a’,1,”string”].

***Função:***

-Deve ser iniciada com a palavra reservada **function**.

Formato: function id (<lista\_parametros>) <retorno>begin <bloco> end

Ex: function teste () begin end

- A lista de parâmetros pode estar fazia ou conter parâmetros no formato: tipo id, tipo id

Ex: (integer a, real b).

- Se a função tiver retorno é necessário colocar “**:**” após o **),** seguido do tipo de retorno.

Ex: function teste (integer a):real begin end

* ***Bloco:***

Contém tudo que pode ser encontrado dentro de um bloco de função.

**Declaração de variáveis:**

A declaração de variáveis locais pode ser feita dentro do bloco e sua atribuição poderá receber, além das mesmas opções das variáveis globais, expressões e chamada funções.

Ex: integer a = b+n;

**Atribuição de variáveis:**

Variáveis locais podem ser atribuídas em qualquer local do bloco. Podendo receber o valor de outra variável, sub-rotina, número, cadeia ou caractere.

Ex: a=1; a=2+3; a= teste(1);

**Write:**

Inicia com a palavra reservada **write**, padrão write (<parâmetros>);

Os parâmetros podem ser cadeias, caracteres, variáveis, números ou expressões aritméticas.

Múltiplas impressões devem ser separadas por vírgula.

**Read:**

Inicia com a palavra reservada **read**, padrão read: read(<parâmetros>);

Múltiplas leituras de variáveis devem ser separadas por vírgula.

Ex: read(a,b,c);

**If:**

Inicia com a palavra reservada **if,** padrão if: if(condição) then begin <bloco> end

Na condição só são permitidas expressões lógicas ou relacionais.

**Else:**

Deve ser antecedido por um bloco if.

Inicia com a palavra reservada **else**, padrão else: else begin <bloco> end

**While :**

Inicia com a palavra reservada **while**, padrão while: while (condição)do begin <bloco> end

Na condição só são permitidas expressões lógicas ou relacionais.

**Chamadas sub-rotinas:**

Inicia com um id(, padrão chamadas sub-rotinas: id(<parâmetros>);

Os parâmetrospodem ser cadeias, caracteres, variáveis, números ou expressões aritméticas.

**Retorno:**

Inicia com id->, padrão retorno: id->(parâmetro);

Os parâmetros do retorno podem ser cadeias, caracteres, variáveis, números ou expressões aritméticas.