

1. O programa irá começar com a palavra *program* e em seguida o nome do seu programa, por exemplo *main* após terá que ser colocado um `;` e após virá o bloco de código, ao fim do programa terá um `.`

Exemplo:

```
program main;  
bloco de código .
```

2. O bloco de código pode conter uma *const* ou um *declaravariaveis* ou *procedure* ou um *corpo*.
 - a. *Tipo Array*: poderá ter 4 tipos sendo eles as 4 primeiras opções mostradas no item 2.b (*integer*, *char*, *string*, *real*) .
 - b. *Tipo*: haverá 5 tipos, eles sendo: *integer* (no qual será numeros inteiros), *char* (no qual será apenas uma caracter dentro de aspas simples), *string*: (no qual será um conjunto de caracteres dentro de aspas duplas), *real* (vai ser decimais com máximo duas casas após o ponto, ex: 1.12), *array*: (terá que iniciar com colchetes, dentro terá que colocar dois números, separados por `..` , este array terá que ter os seguintes tipos declarados no item 2.a) .
 - c. *const*: é uma variável que será constante, poderá ser vazia ou ter um nome, um *tipo*(como explicado no item 2.a) e logo em seguida haverá um `;` , você poderá utilizá-la como dita no item 2.e . E você poderá declarar mais de uma por vez, não precisa reescrever a palavra *const*, e separadas com `;`
Exemplo: `const primeiroNumero=integer; teste=string;`
 - d. *declaravariaveis*: é uma variável que deverá conter um nome, no qual pode-se colocar mais de um nome para declarar várias ao mesmo tempo sendo separadas por um `,` onde após os nomes deverá conter `:` informando qual é o *tipo* da variável (*tipo* está em 2.b) ao fim terá que por `;` . E você poderá declarar mais de uma por vez caso elas sejam do mesmo tipo, não precisa reescrever a palavra *declaravariaveis*, e separadas com uma `,`
Exemplo: `declaravariaveis segundoNumero, terceiroNumero:integer; teste:string;`
 - e. *comandos*: os comandos podem ser *if*, *while*, *repeat*, *read*, *chamaprocedure*, *write*, *for* ou não existir comandos.
 - i. *operadores lógicos*: os operadores lógicos são:

1. `=` : significa igual.
2. `<` : significa menor que.
3. `>` : significa maior que.
4. `>=` : significa maior ou igual que.
5. `<=` : significa menor ou igual que.
6. `<>` : significa diferente que.
7. `or`: significa 'ou'.
8. `and`: significa 'e';

ii. *tipos de expressões*: as expressões podem ser de vários tipos, dentre eles são: os 4 primeiros tipos do item 2.b (*integer*, *char*, *string*, *real*) ou o nome de uma *const* ou *variável*. Também poderá ter os operadores lógicos(2.e.i) como também operadores aritméticos (2.h), por exemplo:

```
1=1 , 2<>1 .
```

iii. *if*: o *if* é como se fosse o “se” (se a flor for uma rosa, ela será bonita) nele irá ter um *[expressão*(os tipos estão no item(2.e.ii) *]* e sequência terá que ter *then* após irá ter *begin* nisso virá o *comando* (2.e) e *end* para sinalizar o fim no *if* e poderá ter um *else* após o bloco do *if* que será explicado no próximo tópico (2.e.iv). Exemplo:

```
if[2>1]then (neste caso se 2 for maior que 1 ele entrará no if)
begin comando de sua escolha (itens 2.e) end
```

iv. *else*: o *else* é o complemento do *if* como se fosse o “se não” (se a flor for uma rosa, ela será bonita, se não ela não será bonita), nele terá que ter *else* após o *begin* nisso virá o *comando* (2.e) e terminará com *end*. exemplo:

```
if[2>1] being comando de sua escolha (itens 2.e) end
else begin comando de sua escolha (itens 2.e) end
```

v. *read*: começará com a palavra ‘*read*’, logo após vira um *parênteses*’()’, e dentro dele haverá um nome de variável, podendo repeti-la separadas com uma vírgula lá dentro. Exemplo:

```
read(segundoNumero)
```

vi. *write*: começara com a palavra ‘*write*’, logo após vira um *parênteses*’()’, e dentro dele haverá um tipo(2.b, no qual poderá ser os 4 primeiros tipos: (*integer*, *string*, *real*, *char*)) ou um texto entre arrobas (@@), poderá repeti-la separadas

com uma vírgula este processo. Exemplo:

```
write(@Hello@, primeironumero)
```

- vii. *comportamentos de loop*: temos três comportamentos do tipo loop, onde nele o código irá ficar lá dentro passando várias vezes até certa condição for atendida, temos três tipos de loop: *repeat*, *for*, *while*.

1. *repeat*: começa com a palavra '*repeat*', logo após vira um comando de sua escolha (itens 2.e) em seguida a palavra '*until*' e após isto um tipo(2.b, no qual poderá ser os 4 primeiros tipos: (*integer*, *string*, *real*, *char*). Exemplo:

```
repeat comando de sua escolha (itens 2.e) until  
[primeiroNumero > segundonumero]
```

2. *for*: Neste loop voce colocara a palavra *for*, logo em seguida você criará um nome de variável, que receberá um tipo(2.b, no qual poderá ser os 4 primeiros tipos: (*integer*, *string*, *real*, *char*), em seguida escreva a palavra '*to*', e em seguida um tipo(2.b, no qual poderá ser os 4 primeiros tipos: (*integer*, *string*, *real*, *char*), após isso escrevera as palavras '*do*', '*begin*', e após isto um comando de sua escolha (itens 2.e) e em seguida a palavra '*end*'. Exemplo:

```
for [item = 0] to [ item < 2] do  
begin comando de sua escolha (itens 2.e) end
```

3. *while*: receberá um tipo(2.b, no qual poderá ser os 4 primeiros tipos: (*integer*, *string*, *real*, *char*), em seguida as palavras '*then*' e '*begin*', em seguida comando de sua escolha (itens 2.e) e logo após a palavra '*end*'.

Exemplo:

```
while[ primeironumero < 2] do  
begin comando de sua escolha (itens 2.e) end
```

- f. *corpo*: o corpo deverá começar sempre com a palavra *begin* e após poderá vir um comando que poderá ser os respectivos citados acima (2.e), após isso virá um `;` e caso queira continuar colocando outros comandos respectivos citados no (2.e) e ao finalizar escrever a palavra *end*; exemplo:

```
begin comando de sua escolha (itens 2.e) end
```

- g. *procedure*: é uma função onde não haverá nenhum retorno, apenas conteúdo dentro dela, nela você deverá colocar um nome como por exemplo *somaNumeros*, também os parâmetros, entre parênteses, se houver. Exemplo:

```
procedure somaNumeros(primeiro:integer)
  declaravariaveis segundo = 2;
begin comando de sua escolha (itens 2.e) end;
```

- h. operadores aritméticos:

1. + : significa soma
2. - : significa subtração
3. * : significa multiplicação
4. / : significa divisão

3. Comentários:

- a. Comentário de linha simples será utilizado // exemplo:

```
// o codigo abaixo será de soma de numeros
```

- b. Comentário de bloco será utilizado no começo /* e ao fim */

exemplo:

```
/* o codigo abaixo será de soma de numeros
que executara variavel 1 + variavel 2 */
```

Regras Léxicas

1. Os dados do tipo *integer* serão aceitos números inteiros de 0 a 9999;
2. Os dados do tipo *real* serão aceitos números inteiros de 0 a 9999 e podendo conter até duas casas decimais com o separador sendo “.” (ponto final);
3. O tipo *char* serão de apenas um caractere e terá que ser envolvido por aspas simples (‘...’);
4. A *string* poderá ter no máximo 5 caracteres e será envolvida por aspas duplas(“...”);
5. O texto que não for nome criado dentro *write* deverá estar envolvido por arrobas (@@);
6. Os nomes de *variáveis* ou *const* não poderão ter espaços, números ou caracteres especiais tendo o tamanho máximo de 5 caracteres, e não poderão ter o mesmo nome das palavras reservadas (*program*, *const*, *procedure* etc) elas são as palavras já definidas.

Erros Léxicos

1. Tipo *integer* for maior que 9999 ou menor que 0;

2. Tipo *real* for maior que 9999 ou menor que 0 ou conter mais que duas casas após o “.” ou conter algum outro caractere especial para separação do decimal;
3. *Char* conter mais de um caractere ou não ser envolvida em aspas simples;
4. *String* conter mais que 5 caracteres ou não ser envolvida em aspas duplas;
5. *Variáveis e const* conterem mais que 5 caracteres como nome ou espaços, números ou caracteres especiais;
6. Comentário de bloco aberto e não fechado;