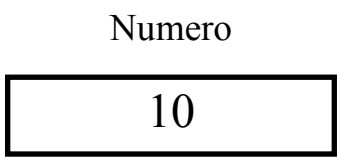
Definição de Pointers

Até agora ao definirmos uma variável, estávamos na verdade alocando um espaço na memória com um tamanho definido pelo tipo da variável (INTEGER, STRING, CHAR, etc...), sendo que ao invés de trabalharmos com o endereço físico de memória temos a facilidade de dar a este espaço alocado um Nome simbólico qualquer.

Exemplo: Alocar na memória para uma variável do tipo INTEGER e atribuir a esta posição de memória um valor qualquer

```
PROGRAM ESTATICO;
VAR
   Número   : INTEGER;
BEGIN
        Número := 10;
END.
```

No Exemplo acima ocorre as seguintes situações:
a) Reservamos espaço na memória suficiente para armazenar dois (2) BYTE's, ou seja, um INTEGER, e demos a esta posição de memória um Nome simbólico: "Número"
b) Atribuímos a variável "Numero" o valor dez (10), o que fará com que a memória ocorra a seguinte situação:



Bem, o que foi mostrado acima é o nosso modo habitual de trabalhar com variáveis, mas a partir de agora iremos trabalhar de uma maneira um pouco diferente, ou seja, ao invés de definirmos uma variável como sendo de um tipo qualquer e a esta variável atribuirmos uma informação propriamente dita, mas sim o endereço físico da memória onde a informação está armazenada. A este tipo de variável passaremos a chamar, a partir de agora, de variáveis pointer (apontadores ou ponteiro), pelo simples fato dela (a variável) apontar, indicar, a localização de uma informação na memória.

Sintaxe para definição:

```
<Nome da variável> : ^<tipo>
```

Exemplo 1: Definir variáveis pointer para os tipos STRING, INTEGER, REAL, CHAR, BOOLEAN.

```
PROGRAM DEFINE_POINTER;
VAR
        Ap_STRING     : ^STRING;
        Ap_INTEGER    : ^INTEGER;
        Ap_REAL       : ^REAL;
        Ap_BYTE       : ^BYTE;
        Ap_CHAR       : ^CHAR;
        Ap_BOOLEAN    : ^BOOLEAN;
BEGIN
        <comandos>;
END.
```

Caso seja necessário definir variáveis pointers para RECORD's e ARRAY's será preciso antes criar tipos de dados que representem estes mesmos RECORD's e ARRAY's.

Exemplo 2: Definir uma variável pointer para um ARRAY[1..2] OF STRING.

```
PROGRAM DEFINE_ARRAY_POINTER;
```

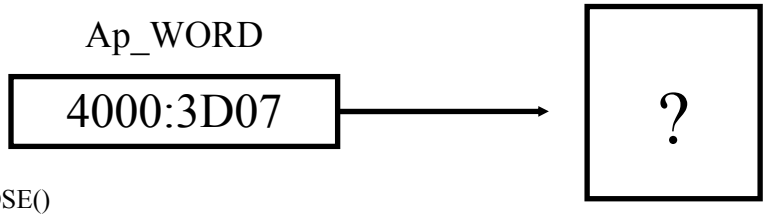
```
TYPE
        Vetor = ARRAY [1..2] OF STRING;
VAR
        Ap_vetor : ^Vetor;
BEGIN
        <comandos>;
END.
```

1.1 Rotinas para Alocação de Memória:

Rotina : NEW()
Função : Aloca espaço na memória para uma informação, com o tamanho definido pelo tipo da variável pointer.
Sintaxe : NEW(Variável Pointer)
Exemplo:

```
PROGRAM ALOCA;
VAR
        Ap_WORD : ^WORD;
BEGIN
        NEW(Ap_WORD);
END.
```

Obs.: No **Exemplo** acima, após o comando NEW, será alocado na memória HEAP, dois BYTE's (uma WORD), sendo que poderemos representar a memória como é mostrado abaixo:



Rotina : DISPOSE()
Função : Libera espaço na memória, o número de BYTE's liberados dependerá do tipo da variável pointer utilizada. Uma vez liberada memória, o valor lá armazenado estará perdido.
Sintaxe : DIPOSE(Variável Pointer)
Exemplo:

```
PROGRAM LIBERA;
VAR
        Ap_WORD : ^WORD;
BEGIN
        NEW(Ap_WORD);
        DISPOSE (Ap_WORD);
END.
```

Obs.: No Exemplo anterior, após o comando DISPOSE, serão liberado dois (2) BYTE's devido ao fato de uma WORD ocupar este espaço de memória.

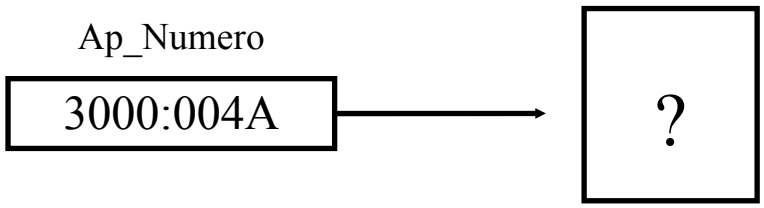
1.2 Atribuição de Valores

A Sintaxe para atribuição de valores é a mesma utilizada em variáveis simples, a única diferença é que devemos colocar após o Nome da variável apontadora o símbolo " ^ "
Exemplo:

```
PROGRAM ATRIBUI;
VAR
        Ap_Número : ^INTEGER;
BEGIN
        NEW(Ap_Número);
        Ap_Número ^:= 10;
        DISPOSE (Ap_Número);
END.
```

No **Exemplo** acima ocorre o seguinte:

a) Criamos uma variável que irá apontar ara dois (2) BYTE's (um INTEGER) na memória.
b) Alocamos espaço suficiente para armazenar um valor do tipo INTEGER e fazemos com que a variável "Ap_Número" aponte para a posição de memória alocada.



c) Colocamos na posição de memória apontada por "Ap_Número" o valor dez (10).
d) Liberamos os dois (2) BYTE's apontados por "Ap_Número". A informação não mais poderá ser acessada.