MSX Article


# *Vetor de tamanho variável*

## Resumo

O objetivo deste artigo é apresentar uma técnica utilizada nos sprites do MSX para utilização de um vetor de tamanho variável, através de strings.

## 1- Introdução

Um vetor é por definição uma lista contendo sempre elementos do mesmo tipo, dispostos sequencialmente em memória, onde o tamanho dessa lista é fixo.

Para entender melhor como funciona um vetor, vejamos primeiro como uma variável Basic é armazenada na memória do MSX [1].

As variáveis são armazenadas em uma área de memória, a partir do endereço apontado por VARTAB (&HF6C2-&HF6C3), terminando no endereço apontado por STREND (&HF6C6-&HF6C7).

Cada variável possui a seguinte estrutura [1]:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| T | N | N | V | V | V | V | V | V | V | V |

Onde:

- T – Tipo de variável:
  - 02 – inteira (2 bytes).
  - 03 – string (3 bytes).
  - 04 – simples precisão (4 bytes).
  - 08 – dupla precisão (8 bytes).
- N – Nome da variável. Pode ser 1 ou 2 caracteres.
- V – Valor. Varia de 2 a 8 bytes.

Cada vez que uma variável é acionada, o interpretador Basic procura pelo seu nome dentro da área de variáveis.

O programa em Basic a seguir ilustra como uma variável é armazenada em memória.

```
10 DEFINT A-B
20 AB=2
30 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
40 PRINT"END  - BYTE - CHAR"
50 PRINT"------------------"
60 FOR F=0 TO 4
70 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
80 NEXT F
```

A variável “AB” do tipo inteiro é criada e armazena o valor 2.

Após executar o programa, a seguinte listagem é exibida:

```
END  - BYTE - CHAR
------------------
80C3 -  02  -
80C4 -  41  -  A
80C5 -  42  -  B
80C6 -  02  -
80C7 -  00  -
```

Uma string possui o seguinte formato:

| T | N | N | C | E | E |
|---|---|---|---|---|---|

Onde:

- T – Tipo de variável. Valor igual a 3 (string).
- N – Nome da variável. Pode ser 1 ou 2 caracteres.
- C – Comprimento da string em bytes.
- E – Endereço do primeiro caractere da string.

O programa em Basic a seguir ilustra como a variável “A$” é armazenada em memória.

```
10 A$="MSX"
20 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
30 PRINT"END  - BYTE - CHAR"
40 PRINT"------------------"
50 FOR F=0 TO 5
60 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
70 NEXT F
80 C=PEEK(P+3)
90 P=PEEK(P+5)*256 + PEEK(P+4)
100 PRINT:PRINT"Lendo a string:":PRINT
110 PRINT"END  - BYTE - CHAR"
120 PRINT"------------------"
130 FOR F=0 TO C-1
140 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
150 NEXT F
```

Saída:

```
END  - BYTE - CHAR
------------------
8199 -  03  -
819A -  41  -  A
819B -  00  -
819C -  03  -
819D -  09  -
819D -  80  -  ♠

Lendo a string:
```

```
END  - BYTE - CHAR
------------------
8009 -  4D  -  M
800A -  53  -  S
800B -  58  -  X
```

Os vetores podem ser de uma dimensão ou mais, no qual se tornam matrizes. Eles são armazenados na memória da seguinte maneira [1]:

| T | N | N | C | C | D | $I_1$ | $I_1$ | ... | $I_d$ | $I_d$ | Dados |
|---|---|---|---|---|---|---|---|---|---|---|---|

Onde:

- T – Tipo de variável.
- N – Nome da variável.
- C – Comprimento final da estrutura, a partir do byte seguinte ao comprimento.
- D – Numero de dimensões.
- $I_1$ – Tamanho da dimensão 1 (mais um do valor da declaração, pois varia de 0 a N).
- $I_d$ – Tamanho da dimensão d (mais um do valor da declaração, pois varia de 0 a N).
- Dados – Dados do vetor ou matriz.

O programa em Basic a seguir ilustra a matriz “VT” na memória.

```
10 DEFINT V
20 DIM VT(2,1)
30 VT(0,0)=1:VT(1,0)=2:VT(2,0)=3
40 VT(0,1)=4:VT(1,1)=5:VT(2,1)=6
50 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
60 PRINT"END  - BYTE - CHAR"
70 PRINT"------------------"
80 FOR F=0 TO 43
90 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
100 NEXT F
```

Saída:

```
END  - BYTE - CHAR
------------------
8123 -  02  -
8124 -  56  -  V
8125 -  54  -  T
8126 -  11  -
8127 -  00  -
8128 -  02  -
8129 -  02  -
812A -  00  -
812B -  03  -
812C -  00  -
812D -  01  -
```

```
812E -  00  -
812F -  02  -
8130 -  00  -
8131 -  03  -
8132 -  00  -
8133 -  04  -
8134 -  00  -
8135 -  05  -
8136 -  00  -
8137 -  06  -
8138 -  00  -
```

O comprimento possui 17 bytes (&H11), variando do número de dimensões até o final dos dados. É contabilizado da seguinte forma:

comprimento = d + $I_1$ + $I_2$ + dados

Para achar o comprimento dos dados, tem-se:

17 = 1 + 2 + 2 + dados
dados = 17 - 5 = 12

Como cada elemento inteiro possui 2 bytes, tem-se que dados possui 12/2 ou 6 elementos.

## 2- Modificando o tamanho de vetores

Um vetor não pode ter seu tamanho alterado. Mas, caso isso seja necessário, pode-se criar um novo vetor com a dimensão desejada e copiar os valores do vetor antigo para o novo. Então, apagar o vetor antigo através do comando Basic “erase” seguido do nome da variável. Ex: ERASE VT.

Uma alternativa para se ter um vetor de tamanho variável é a utilização de strings. As strings são cadeia de caracteres e possuem comprimento variável, conforme foi visto no capítulo anterior. Além disso, as strings possuem algumas funções em Basic criadas para manipulá-las.

As funções para manipular strings são:

- LEFT$(S, N) – seleciona uma cadeia de caracteres de comprimento “N” da string “S”, a partir da esquerda.
- RIGHT$(S, N) – seleciona uma cadeia de caracteres de comprimento “N” da string “S”, a partir da direita.
- MID$(S,P,N) – seleciona uma cadeia de caracteres de comprimento “N” da string “S”, a partir da posição “P”, contada da esquerda para a direita.
- LEN(S) – Retorna o comprimento da string “S”.

Além disso, é possível concatenar (juntar, adicionar) caracteres ou strings a uma string existente através do operador “+”. Ex:

```
10 A$ = “ABC”
20 A$ = A$ + “D”
```

Assim, a variável “A$” passa a conter a string “ABCD”.

Para retirar o último elemento da string:

```
30 A$ = LEFT$(A$, LEN(A$)-1)
```

A principal vantagem do uso de strings é que ela sempre será uma lista com o tamanho necessário de elementos. Entretanto, as strings só permitem dados de um tipo: caractere. Os caracteres possuem 1 byte para armazenar dados. Dessa forma, nossa lista só poderá armazenar números inteiros de um byte (ou caracteres).

Na atribuição de uma string, só é permitido fornecer caracteres a ela. Dessa forma, é necessário utilizar a função STR$(val) – que converte de byte para caractere – para inserir dados na string.

Suponhamos uma lista com 10 números inteiros, com elementos variando de 1 a 10. O programa em Basic a seguir irá armazenar essa lista em uma string.

```
10 FOR F=1 TO 10
20 L$ = L$ + CHR$(F)
30 NEXT F
```

Para recuperar um elemento, na posição P dessa lista, vamos utilizar a função MID$. Além disso, o caractere precisa ser convertido de volta para byte, através da função ASC.

```
10 FOR F=1 TO 10
20 L$ = L$ + CHR$(F)
30 NEXT F
40 FOR P=1 TO 10
50 V = ASC(MID$(L$, P, 1))
60 PRINT V
70 NEXT P
```

## 3- O truque aplicado aos sprites [2]

A instrução em Basic “SPRITE$(n) = string” define o desenho que irá aparecer na tela. A variável “n” é uma identificação para o sprite, que varia de 0 a 255 no modo 8x8, e de 0 a 63 no modo 16x16. Já “string” é uma cadeia de caracteres que irá conter a configuração de desenho do sprite.

Cada caractere da string de definição do sprite é um código ASCII que irá conter a informação de pixel aceso e pixel apagado do sprite. Ao converter-se esse valor do código ASCII para binário, os valores de bit iguais a um irão acender o pixel do sprite, enquanto que os valores de bit igual a zero irão apagar.

No modo 8x8, cada caractere define uma linha, onde cada bit do código ASCII define uma coluna da tela. O exemplo a seguir contém a configuração para desenhar a letra “M” no modo 8x8:

```
A$ = CHR$(&b01000010)
B$ = CHR$(&b01100110)
C$ = CHR$(&b01011010)
D$ = CHR$(&b01011010)
E$ = CHR$(&b01000010)
F$ = CHR$(&b01000010)
G$ = CHR$(&b01000010)
H$ = CHR$(&b01000010)

SPRITE$(1) = A$+B$+C$+D$+E$+F$+G$+H$
```

Note que os bits 0 e 1 definem a forma do desenho. Os valores igual a “1” foram destacados em vermelho para melhor visualizar o formato da letra “M”.

O sprite no modo 8x8 possui a string com comprimento igual a 8. Já o sprite no modo 16x16 terá comprimento da string igual a 32.

## 4- Créditos

Este artigo foi escrito por Marcelo Silveira, em Janeiro de 2017.
E-mail: flamar98@hotmail.com

Referências:

[1] – MSX Top Secret, Edison Moraes, disponível em: http://www.msxtop.msxall.com.

[2] – Artigo: “Sprites e Gravidade”, Marcelo Silveira, disponível em: http://marmsx.msxall.com .