MSX Article

Agenda EXPERTa

Resumo

O objetivo deste artigo é modificar o programa “Agenda EXPERTa”, publicado no manual do Expert MSX da Gradiente, de forma a ler e gravar os dados da agenda em disco.

1- Introdução

O programa “Agenda EXPERTa”, publicado no manual do Expert nos anos 80 [1], tem como objetivo ler e armazenar o nome e o telefone de um determinado número de pessoas na memória do MSX, de forma a ilustrar a criação e o uso de matrizes no Basic do MSX.

O programa da “Agenda EXPERTa” é apresentado a seguir.

Agenda1.bas

```
10 REM Agenda EXPERTa
20 PRINT:PRINT:PRINT"Consultar (C) ou  Inserir (I)"; : POKE &HFCAB,255
30 INPUT S$
40 IF S$="C" THEN GOTO 160
50 IF S$="I" THEN GOTO 70
60 GOTO 30
70 PRINT"Quantos nomes serão inseridos";
80 INPUT N : IF A>0 THEN ERASE N$
90 A=1 : DIM N$(N,2)
100 FOR F=1 TO N
110 PRINT "Entre o nome";F;":"
120 INPUT N$(F,1)
130 PRINT "Entre o telefone:"
140 INPUT N$(F,2)
150 NEXT F:GOTO10
160 PRINT:PRINT "Nome","Telefone"
170 PRINT
180 FOR F=1 TO N
190 PRINT N$(F,1),N$(F,2)
200 NEXT F
300 GOTO 10
```

Menu

Poke para travar letras maíusculas

Sub-rotina para inserir contatos na agenda

Sub-rotina para imprimir os contatos da agenda na tela

2- Matrizes e Arquivos

2.1- Vetores e Matrizes

Uma variável armazena na memória do computador um determinado tipo de dado. Os tipos de dados variam de acordo com a linguagem utilizada. No caso do Basic, são dois tipos principais: numérico e string. O tipo numérico ainda se divide em inteiro, precisão simples e precisão dupla.

Cada variável pode somente armazenar um, e somente um, dado do tipo definido para ela. No caso do programa da agenda, se fossem inseridos 100 nomes e telefones, teriam que ser criadas 100 variáveis para armazenar os nomes e mais 100 para armazenar os telefones. Isso seria inviável em termos de programação e gerenciamento do programa.

A solução para esse problema é a criação de um vetor, pois é um tipo de variável que pode armazenar vários dados da mesma natureza. Assim, ao referenciar um dos 100 nomes ou telefones, essa referencia seria feita ao mesmo nome de variável.

A figura 1 ilustra a diferença entre uma variável *V* do tipo numérico e um vetor *V(n)* também do tipo numérico.

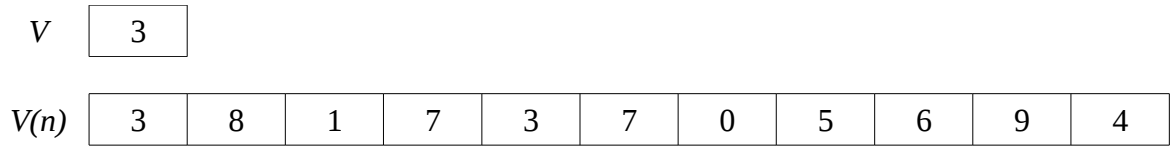

Figura 1. Diferença entre uma variável comum e um vetor.

As variáveis da figura 1 poderiam indicar a venda diária de televisores de uma loja de departamentos. Enquanto que *V* é capaz de somente registrar a venda de um dia, *V(n)* pode registrar a venda de 11 dias.

No caso da agenda, a natureza dos dados está diretamente relacionada a uma pessoa. Entretanto, cada pessoa deve possuir informações quanto ao nome e ao telefone. Assim, os vetores são insuficientes armazenar os dados da agenda, uma vez que ele possui apenas uma dimensão e daria somente para armazenar o nome ou o telefone.

Para resolver esse problema, deve-se utilizar uma matriz, que é um vetor de duas ou mais dimensões. Nesse caso, seria criada uma tabela, onde cada linha representa uma pessoa e as colunas os dados relativos a essa pessoa, como o nome e telefone.

Os vetores ou matrizes são criados em Basic, através do comando DIM, seguido do nome da variável e o tamanho entre parêntesis. O número de argumentos (parâmetros) passados na criação de uma matriz irá indicar o número de dimensões da matriz.

Exemplos:

- DIM V(10) - cria um vetor numérico com 11 posições (0 a 10).
- DIM M$(20,4) - cria uma tabela de strings com 21 linhas e 5 colunas (0 a 20, 0 a 4).
- DIM C(5,5,5) - cria um cubo de valores numéricos de lado igual a 6.

O Basic do MSX cria um vetor ou matriz sempre variando de 0 a N. Entretanto, o programa da agenda utiliza a tabela da posição 1 a N, descartando a posição 0.

O uso dos parêntesis após o nome da variável indica que esta variável é um vetor ou matriz. O valor contido entre os parêntesis é a posição de um dado que se deseja alterar ou consultar. Observe na figura 2, o vetor da figura 1 com os índices *n* assinalados.

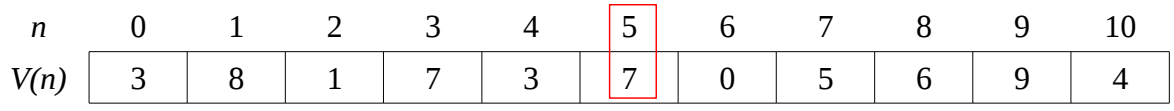

Figura 2. O vetor e seus índices.

Assim, o comando em Basic *PRINT V(5)* imprime na tela o valor da posição 5 do vetor *V*, que é 7. No caso da matriz *M$*, o comando *PRINT M$(1,4)* imprime a string da linha 1 e coluna 4 da matriz.

Conforme visto, o tamanho do vetor ou matriz é definido no ato da criação e não pode ser mudado. Assim, o programa da agenda pergunta ao usuário quantos nomes (pessoas) serão inseridos na agenda antes de começar. Outra coisa que se deve ter em mente é que uma matriz deve ter todos os dados do mesmo tipo. Assim, a matriz da agenda é de strings.

Supondo a agenda criada e preenchida com três pessoas, essa matriz (*N$)* é representada na figura 3.

	0	1	2
0			
1		CARLA	1234-5678
2		SUELI	1234-9876
3		MARCOS	1243-7766

Figura 3. A tabela da agenda.

O comando *PRINT N$(1,1)* imprime a string “CARLA”.

2.2- Arquivos

O programa da agenda guarda os dados na memória principal do MSX. Essa memória é volátil e, quando o MSX é desligado, os dados são perdidos.

Na época de lançamento do MSX 1, os disk-drives eram raridade. Entretanto, com o popularização dos disk-drives alguns anos depois, tornou-se então possível gravar os dados no disquete em arquivos , para posteriormente recuperá-los, mesmo com o desligamento do MSX.

Os arquivos em Basic do MSX possuem os seguintes formatos:

- Binário, através dos comandos BSAVE/BLOAD.
- Texto, através dos comandos OPEN/CLOSE.

Os comandos BSAVE e BLOAD trabalham com blocos de memória e não são apropriados para o acesso a dados em variáveis. De forma distinta, os comandos OPEN e CLOSE permitem a manipulação de dados em variáveis e matrizes, sendo assim, o mais apropriado ao programa da agenda.

Assim como em outras linguagens, os comandos OPEN/CLOSE possuem o seguinte fluxo de funcionamento:

- Abrir o programa para leitura ou gravação (OPEN)
- Ler ou escrever dados no disco (INPUT ou PRINT)
- Fechar o arquivo (CLOSE)

A sintaxe para o comando OPEN é:

```
OPEN <nome_do_arquivo> FOR INPUT AS #n    ' Leitura
OPEN <nome_do_arquivo> FOR OUTPUT AS #n   ' Gravacao
```

Onde *n* é um número que representa um arquivo aberto, e que pode variar de 0 a 15. Entretanto, o MSX somente é capaz de gerenciar 6 arquivos de disco ao mesmo tempo [2]. O número de arquivo *n* é o equivalente ao ponteiro de arquivo em Pascal e C.

Exemplo:

```
OPEN "DADOS.DAT" FOR OUTPUT AS #1
```

A leitura dos dados é feita através do comando *INPUT #n* ou *LINE INPUT #n*, seguido da variável do tipo string, no qual é passado o conteúdo de uma linha inteira. No caso do *INPUT*, a leitura da linha é interrompida, quando um caractere delimitador é encontrado (ex: vírgula). Ex:

```
INPUT #1, A$
PRINT A$
```

Ao ler um dado, se o arquivo não foi aberto pelo OPEN, uma mensagem de erro surge.

A gravação de dados é feita através do comando *PRINT #n*. Esse comando possui a mesma sintaxe do comando PRINT, ou seja, passa automaticamente para a próxima linha ou permanece na mesma linha, ao acrescentar o ponto e vírgula ao final da expressão. Ex:

```
PRINT #1, A$
```

O comando CLOSE serve para fechar o arquivo aberto. Ex:

```
CLOSE #1
```

Embora os comandos OPEN e CLOSE do Basic criem sempre arquivos no formato de texto, existem diversas formas de se armazenar os dados em um arquivo. O curso de Basic da página MarMSX Development [3] apresenta em detalhes esses formatos e que podem ser utilizados para a agenda. Entretanto, será adotado nesse artigo o formato de dados mais simples.

3- Adicionando o recurso de leitura/gravação em disco

O programa original da agenda "Agenda1.bas" será modificado, de forma a ler e gravar os dados armazenados na tabela de nomes. Algumas outras modificações foram introduzidas, como a reformulação do menu principal e a adição de comentários para melhorar a visualização e identificação das sub-rotinas.

Foi utilizado um nome de arquivo padrão para ler e gravar dados da agenda, chamado de AGENDA.DAT. Conforme visto na seção anterior, é um arquivo no formato de texto.

O programa "Agenda2.bas" acrescenta duas rotinas ao programa original: uma para gravação dos dados da tabela em disco, outra para a recuperação dos dados.

Agenda2.bas

```
10 REM Agenda EXPERTa
15 POKE &HFCAB,255
20 CLS:PRINT"Agenda EXPERTa":PRINT:PRINT"Consultar (C)":PRINT"Inserir
(I)":PRINT"Gravar (G)":PRINT"Ler (L)":PRINT
30 INPUT S$
40 IF S$="C" THEN GOSUB 160
50 IF S$="I" THEN GOSUB 70
55 IF S$="G" THEN GOSUB 300
56 IF S$="L" THEN GOSUB 400
60 GOTO 20
67 '
68 ' Insere
69 '
70 PRINT"Quantos nomes serão inseridos";
80 INPUT N : IF A>0 THEN ERASE N$
90 A=1 : DIM N$(N,2)
100 FOR F=1 TO N
110 PRINT "Entre o nome";F;":"
120 INPUT N$(F,1)
130 PRINT "Entre o telefone:"
140 INPUT N$(F,2)
150 NEXT F: RETURN
157 '
158 ' Consulta
159 '
160 CLS:PRINT:PRINT "Nome","Telefone"
170 PRINT
180 FOR F=1 TO N
190 PRINT N$(F,1),N$(F,2)
200 NEXT F
210 A$ = INPUT$(1) : RETURN
297 '
298 ' Grava
299 '
300 OPEN "AGENDA.DAT" FOR OUTPUT AS#1
310 FOR I=1 TO N
320 PRINT#1, N$(I,1)
325 PRINT#1, N$(I,2)
330 NEXT I
340 CLOSE #1
350 RETURN
397 '
398 ' Le
399 '
400 OPEN "AGENDA.DAT" FOR INPUT AS#1
410 IF A>0 THEN ERASE N$: A=1 : N=0
420 INPUT#1, DM$
430 N = N+1
440 IF NOT EOF(1) THEN 420
450 N = N/2 : CLOSE#1
460 DIM N$(N,2)
470 OPEN "AGENDA.DAT" FOR INPUT AS#1
480 FOR I=1 TO N
490 INPUT #1, N$(I,1)
500 INPUT #1, N$(I,2)
510 NEXT I
520 CLOSE#1 : RETURN
```

A sub-rotina de gravação realiza os seguintes passos:

1. Abrir o arquivo AGENDA.DAT para gravação.
2. Varrer a tabela utilizada como agenda, gravando para cada contato, o nome em uma linha e o telefone na linha seguinte.
3. Fechar o arquivo.

O arquivo AGENDA.DAT possui a seguinte formatação:

```
NOME1
TELEFONE1
NOME2
TELEFONE2
…
NOMEn
TELEFONEn
<fim_de_arquivo>
```

A sub-rotina de leitura realiza os seguintes passos:

1. Abrir o arquivo AGENDA.DAT para leitura.
2. Varrer o arquivo, lendo linha a linha, de modo a contar o número de contatos gravados.
3. Como cada contato ocupa 2 linhas, faz-se N = N/2
4. Fechar o arquivo.
5. Destruir a tabela antiga, se existente, e criar nova tabela com o tamanho calculado.
6. Abrir o arquivo AGENDA.DAT para leitura.
7. Agora, os dados são lidos do arquivo e armazenados na tabela.
8. Fechar arquivo.

3.1- Corrigindo um pequeno incoveniente

O arquivo AGENDA.DAT não possui a informação da quantidade de contatos existentes, forçando a sub-rotina de leitura do arquivo a contá-los antes de criar a tabela para armazenar os nomes e os telefones.

A solução para este problema é simples. Basta informar na primeira linha do arquivo AGENDA.DAT a quantidade de contatos existentes nesse arquivo. Assim, ao ler essa linha, o programa irá saber a quantidade total de contatos e não precisará mais varrer duas vezes o arquivo para recuperar os dados.

O novo formato do arquivo AGENDA.DAT:

```
NUM_REGISTROS
NOME1
TELEFONE1
NOME2
TELEFONE2
…
NOMEn
TELEFONEn
<fim_de_arquivo>
```

O programa “Agenda3.bas” corrige o problema e lê o novo formato de arquivo.

Agenda3.bas

```
10 REM Agenda EXPERTa
15 POKE &HFCAB,255
20 CLS:PRINT"Agenda   EXPERTa":PRINT:PRINT"Consultar   (C)":PRINT"Inserir
(I)":PRINT"Gravar (G)":PRINT"Ler (L)":PRINT
30 INPUT S$
40 IF S$="C" THEN GOSUB 160
50 IF S$="I" THEN GOSUB 70
55 IF S$="G" THEN GOSUB 300
56 IF S$="L" THEN GOSUB 400
60 GOTO 20
67 '
68 ' Insere
69 '
70 PRINT"Quantos nomes serão inseridos";
80 INPUT N : IF A>0 THEN ERASE N$
90 A=1 : DIM N$(N,2)
100 FOR F=1 TO N
110 PRINT "Entre o nome";F;":"
120 INPUT N$(F,1)
130 PRINT "Entre o telefone:"
140 INPUT N$(F,2)
150 NEXT F: RETURN
157 '
158 ' Consulta
159 '
160 CLS:PRINT:PRINT "Nome","Telefone"
170 PRINT
180 FOR F=1 TO N
190 PRINT N$(F,1),N$(F,2)
200 NEXT F
210 A$ = INPUT$(1) : RETURN
297 '
298 ' Grava
299 '
300 OPEN "AGENDA.DAT" FOR OUTPUT AS#1
310 PRINT#1, N
320 FOR I=1 TO N
330 PRINT#1, N$(I,1)
340 PRINT#1, N$(I,2)
350 NEXT I
360 CLOSE #1
370 RETURN
397 '
398 ' Le
399 '
400 OPEN "AGENDA.DAT" FOR INPUT AS#1
410 IF A>0 THEN ERASE N$: A=1
420 INPUT#1, N
430 DIM N$(N,2)
440 FOR I=1 TO N
450 INPUT #1, N$(I,1)
460 INPUT #1, N$(I,2)
470 NEXT I
480 CLOSE#1 : RETURN
```

Um exemplo de arquivo gerado para 3 contatos:

```
 3
CARLA
1234-5678
SUELI
1234-9876
MARCOS
1243-7766
```

4- Créditos

Este artigo foi escrito por Marcelo Silveira, em Outubro de 2016.

Revisado em: julho de 2017.
e-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com

Referências:
[1] - Livro: Dominando o Expert, editora Aleph, 5a. Edição, 1987.
[2] - Manual de instruções do Drive Interface DDX.
[3] - Curso de Basic, MarMSX, em http://marmsx.msxall.com.