MSX Article

Funções em Basic

Resumo

O objetivo deste artigo é mostrar como utilizar o recurso de funções em uma linguagem não estruturada como o Basic.

1- Introdução

Uma função é uma sub-rotina destacada do programa principal, com o objetivo de resolver um determinado problema recorrente, evitando assim a duplicidade de código. Uma função geralmente recebe um conjunto de parâmetros, realiza determinadas operações e retorna um valor como resposta.

Em linguagens estruturadas como C e Pascal, a função é um trecho de código explicitamente separado do corpo principal do programa. Nessas linguagens, uma função é composta do nome da função, dos parâmetros de entrada, do tipo de dado de retorno e do código de tratamento da função. Ex:

```
tipo_de_retorno nome_da_função(lista de parâmetros)
{
     // código da função
}
```

Como a linguagem Basic é não-estruturada, as funções serão construídas logicamente em seu corpo principal.

Em Basic, temos as funções nativas da linguagem e as criadas pelo usuário. Como exemplo de funções nativas, temos:

`abs(arg)`	`atn(arg)`	`bin$(arg)`	`hex$(arg)`
`peek(arg)`	`poke(arg)`	`sin(arg)`	`right$(args)`

Você saberia identificar em uma função do Basic o tipo de retorno?

É simples. Quando o nome da função possui a terminação em dólar “$”, o retorno da função é alfanumérica (string). Quando não possui, o retorno é numérico. Dessa forma, ao analisarmos os exemplos acima, concluímos que a função “abs” retorna um valor numérico, enquanto que a função “hex$” retorna um valor alfanumérico.

Ao chamarmos uma função em Basic, é obrigatório o armazenamento do resultado, ou seja, o interpretador Basic não aceita o uso da função “solta”. Isso pode ser feito de duas maneiras: ou através de uma variável do mesmo tipo de retorno ou do comando print. Ex:

`i = abs(-1)`	✔	`n$ = bin$(10)`	✔
`print abs(2)`	✔	`print bin$(10)`	✔
`abs(2)`	✖	`bin$(10)`	✖
`c$ = abs(-1)`	✖	`f = bin$(10)`	✖

2- Funções do usuário: DEF FN

O usuário também pode criar funções no Basic. Isso é feito de duas maneiras:

- Funções simples através do comando DEF FN.
- Funções mais complexas com o uso de sub-rotinas (GOSUB-RETURN).

O comando DEF FN cria uma função nos moldes das funções nativas. Entretanto, ela comporta somente expressões e não sub-rotinas. A sua sintaxe é:

```
DEF FN nome_da_função (lista_de_parâmetros) = expressão
```

O interpretador Basic não aceita a definição de uma função como um comando direto (*illegal direct*). Dessa forma, devemos colocá-la sempre dentro de um programa.

No primeiro programa exemplo, vamos criar uma função chamada “quad”, que retorna o quadrado de um número.

```
10 DEF FN QUAD(X) = X*X
```

Como chamar a função criada? O uso de uma função definida pelo usuário é feito através do nome da função, precedido obrigatoriamente por FN, e dos parâmetros. Ex:

```
20 PRINT FN QUAD(4)
```

Tipo de dado do parâmetro obrigatoriamente igual ao definido.

Programa completo:

exemplo1.bas

```
10 DEF FN QUAD(X) = X*X
20 PRINT FN QUAD(4)
```

Saída:
16

A expressão da função pode chamar outras funções também. Exemplo:

```
10 DEF FN CALC(X) = X*SIN(X)
```

No exemplo a seguir, vamos utilizar três parâmetros de entrada.

exemplo2.bas

```
10 DEF FN FUNXYZ(X,Y,Z) = (X+Y)*Z
20 PRINT FN FUNXYZ(1,2,3)
```

Saída:
9

A lista de parâmetros segue rigorosamente a ordem da definição. Assim, X recebe o valor 1, Y o valor 2 e Z o valor 3.

No próximo exemplo, vamos fazer a concatenação de uma dada string com ela mesmo.

exemplo3.bas

```
10 DEF FN CAT$(S$) = S$+S$
20 PRINT FN CAT$("BE")
```

Saída:
BEBE

É permitido também definir a lista de parâmetros da função com diferentes de tipos de dado. Ex:

```
10 DEF FN F(S$, C) = LEN(LEFT$(S$,C)+RIGHT$(S$,C))
```

Do mesmo modo, podemos ter um tipo de dado no parâmetro e outro no retorno, conforme o exemplo a seguir.

Agora vamos criar uma função para formatar a saída de um número com o valor zero precedido antes, caso ele possua uma quantidade de casas menor que um determinado tamanho. Nesse primeiro exemplo, vamos admitir que o número formatado tenha 2 casas decimais.

exemplo4.bas

```
10 DEF FN FRMT$(N) = RIGHT$("0"+RIGHT$(STR$(N),LEN(STR$(N))-1),2)
20 PRINT FN FRMT$(4)
30 PRINT FN FRMT$(10)
```

Saída:
04
10

Agora, o número de casas decimais é passado como o parâmetro “C”.

exemplo5.bas

```
10 DEF FN FRMT$(N,C) = RIGHT$(STRING$(C-1,"0")+RIGHT$(STR$(N),LEN(STR$
(N))-1),C)
20 PRINT FN FRMT$(5,3)
30 PRINT FN FRMT$(32,5)
```

Saída:
005
00032

Para finalizarmos o uso do DEF FN, vamos formatar a saída dos dados de uma tabela, utilizando o print using e a função do “exemplo4.bas” para formatar a data de nascimento.

exemplo6.bas

```
10 DEF FN FRMT$(N) = RIGHT$("0"+RIGHT$(STR$(N),LEN(STR$(N))-1),2)
20 DIM T$(2):DIM NA(2,3)
30 T$(1) = "BEATRIZ":NA(1,1)=4:NA(1,2)=6:NA(1,3)=1985
40 T$(2) = "GERSON":NA(2,1)=25:NA(2,2)=2:NA(2,3)=1988
50 FOR I=1 TO 2
60 PRINT USING "NOME: \       \ - DATA_NASC: &/&/####";T$(I);FN FRMT$
(NA(I,1));FN FRMT$(NA(I,2));NA(I,3)
70 NEXT I
```

Saída:
NOME: BEATRIZ – DATA_NASC: 04/06/1985
NOME: GERSON – DATA_NASC: 25/02/1988

3- Funções do usuário: GOSUB-RETURN

Podemos criar funções também utilizando o recurso de GOSUB-RETURN. Diferente do DEF FN, agora é possível criar uma sub-rotina para o tratamento de uma função. Entretanto, não podemos passar parâmetros e tampouco receber o retorno da função. Nesse caso, é necessário o uso de variáveis globais para tal fim.

O “exemplo1.bas” da seção anterior calcula o quadrado de um número. Vamos criar a mesma função através do GOSUB-RETURN.

exemplo7.bas

```
10 N=4
20 GOSUB 500
30 PRINT "O quadrado de ";N;"e': ";Q
40 END
500 REM
510 REM FUNCAO QUAD
520 REM
530 Q = N*N
540 RETURN
```

Saída:
O quadrado de 4 e': 16

Conforme dito na seção 1, as funções em Basic são criadas logicamente no programa. Para ficar mais destacada do restante do código, colocamos a função em uma numeração mais avançada, bem como sinalizamos o inicio dela com comentários, de forma que seja facilmente localizada.

A função GOSUB-RETURN equivalente ao “exemplo4.bas” é a seguinte:

exemplo8.bas

```
10 N=4
20 GOSUB 500
30 PRINT NF$
40 END
500 REM
510 REM FUNCAO FRMT
520 REM
530 NF$ = STR$(N)
540 NF$ = RIGHT$("0"+RIGHT$(NF$,LEN(NF$)-1),2)
550 RETURN
```

Saída:
04

4- Créditos

Este artigo foi escrito por Marcelo Silveira, em Abril de 2017.

Home-page: http://marmsx.msxall.com
E-mail: flamar98@hotmail.com