**Passando Arrays Como Parâmetros em C**

**Mário Leite**

**...**

Já fiz aqui uma comparação entre as funções em **C** e as fábricas; a matéria prima para a fábrica é o equivalente aos parâmetros para as funções, e podem ser externos (*recebidos de outra entidade*) ou como dados internos (*obtidos na própria rotina*). Para as fábricas, ou para as funções, essa “matéria prima” pode ser várias: separadamente, ou reunidas em um só ”pacote”. No caso das funções em **C** esse “pacote” de parâmetros é representado por um *array* (*matriz* ou *vetor*) pelo conceito de ponteiros, que simulam uma passagem de parâmetros “*por referência*”. No caso de parâmetros matriciais unidimensionais (*vetores*) a **figura 1** mostra três situações em que isto pode ocorrer. A **figura 2** mostra um exemplo de programa para o caso de vetor não dimensionado, passado para uma função e recebendo de volta o maior elemento desse vetor; e a **figura 2.1** mostra a saída do programa.

Agora, uma questão que pode ser levantada: “*Como a função chamadora, no caso* ***main()****, poderia receber todo o vetor, e não apenas um só elemento?!*” Muito simples; basta dimensionar o vetor no retorno da função, e pronto: a função *chamadora* receberá o vetor como uma variável “empacotada”. Observe o código na **figura 3** e sua saída na **figura 3.1**. E neste caso, para exibir o vetor ordenado pela função, a função ***main*()** não pegou o retorno de **OrdenaVetor**(); apenas *exibiu* seus elementos, aproveitando-se do fato da passagem de parâmetros ter sido feita “*por* referência”.

Para passar uma matriz (*array bidimensional*) a primeira dimensão não deve ser passada explicitamente; isto tem a ver com o fato de que matrizes exigem muito tempo de processamento e “enchem” a memória do computador; e como a primeira dimensão indica (aponta) o local de início da matriz, basta esse ponto para que ela seja alocada, reduzindo o tempo de processamento. O programa ilustrado na **figura 4** mostra essa situação, onde a função **RecebeMat**() recebe uma matriz **M** passada como parâmetro.

-----------------------------------------------------------------------------------------------------------------------------------

**int** main()

{

**int** V[8], R;

...

R **=** Func**(**V**)** *//chama a função e passa-lhe o vetor como parâmetro*

...

}

----------------------------------------------------------------

**int** Func(**int** \*x) *//função recebe parâmetro como ponteiro*

}

...

...

}

----------------------------------------------------------------

**int** Func(**int** x[8]) *//recebe parâmetro como vetor dimensionado*

}

...

...

}

----------------------------------------------------------------

**int** Func(**int** x[]) *//recebe parâmetro como vetor não dimensionado*

}

...

...

}

**Figura 1 - *Layouts* de três tipos funções que recebem um vetor como parâmetro**

//Passa vetor como parâmetro e recebe o maior elemento

//Em C

//Autor: Mário Leite

#include <stdio.h>

#include <conio.h>

**int** VerifMaior(**int** V[]); *//protótipo da função*

**int** main()**{**

**int** Vet[8] = {4, 5, 8, 17, 2, 6, 1, 0};

**int** Maior;

Maior= **VerifMaior**(Vet)**;** *//chama a função*

printf("Maior elemento do vetor: %d", Maior);

getch();

**return** 0;

**}**

//**-----------------------------------------------------------**

**int** VerifMaior(**int** V[]) *//com parâmetro não dimensionado*

**{**

**int** i, j, aux, M;

*//Usa o* "*Método da Bolha*" *para ordenar o vetor*

**for**(i=0;i<=7;i++)

**{**

**for** (j=i+1; j<=8; j++)

**{**

**if**(V[i]>V[j])

**{**

aux = V[i];

V[i] = V[j];

V[j] = aux;

**}**

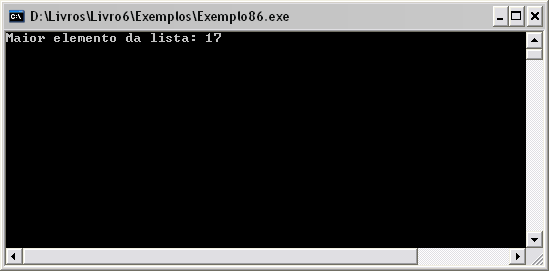
**}**

**}**

**return** V[7]; *//O maior elemento é o último (ordenado)*

*}*

**Figura 2 - Código do programa para exibir o maior elemento de um vetor**



**Figura 2.1 - Saída do programa da Figura 2**

//Passa vetor como parâmetro e exibe o vetor ordenado

//Autor: Mário Leite

#include <stdio.h>

#include <conio.h>

**int** OrdenaVetor(**int** V[]); *//protótipo da função*

**int** main()**{**

**int** Vet[8];

**int** j, k, Maior;

**for** (j=0; j<8; j++)

**{**

k = j + 1;

printf("Entre com o elemento %d do vetor: ", k);

scanf("%d", &Vet[j]);

**}**

printf("\n\n");

printf("Vetor ordenado \n");

**OrdenaVetor**(Vet); *//chama a função e passa vetor como parâmetro*

*//Imprime o vetor (ordenado) recebido como retorno da função*

**for** (j=0; j<8; j++)

**{**

printf("%d %s", Vet[j], " ");

**}**

getch();

**return** 0;

**}**

*//Implementação da função* ***OrdenaVetor****()*

**int** OrdenaVetor(**int** V[])

**{**

**int** i, j, aux;

**for** (i=0;i<7;i++)**{**

**for** (j=i+1; j<8; j++)**{**

**if**(V[i]>V[j]){

aux = V[i];

V[i] = V[j];

V[j] = aux;

**}**

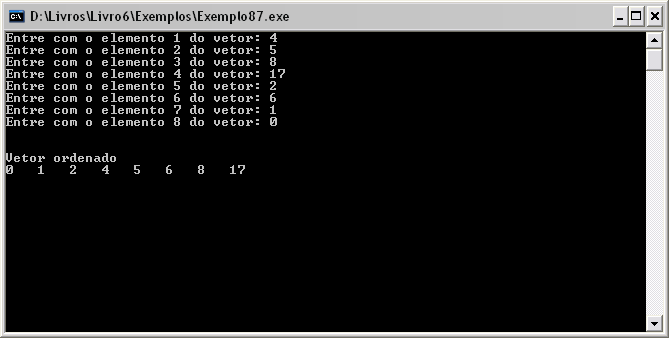
**}**

**}**

**return** V[7]; *//retorno da função com o vetor ordenado*

**}**

**Figura 3 - Código do programa para exibir um vetor ordenado por uma função**



**Figura 3.1 - Saída do programa da Figura 3**

**int** RecebeMat(**int** M[][5], **int** a); *//protótipo da função*

**int** main()

**{**

...

**int** A[2][5] = {{1, 2, 3, 4, 5},

{6, 7, 8, 9, 0}};

X = **RecebeMat**(A,7); *//chama a função*

**return** 0

**}**

*//Implementação da função*

**int** RecebeMat(**int** M[][5], **int** a)

**{**

...

...

**return** valor;

**}**

**Figura 4 - Programa para passar uma matriz como parâmetro**