


Vetores e Matrizes como argumentos de funções

Vetores

Vetores podem ser passados também como argumentos de funções.

Vamos escrever uma função que **receba um vetor** e retorna a sua **média**.



```
int media(int valor[], int nElementos)
{
    int i, soma = 0;
    for(i = 0; i < nElementos; i++)
        soma=soma + valor[i];

    return soma / nElementos;
}
```

Note que na declaração da função, o argumento que representa o **vetor** é declarado com colchetes. Além dele, passamos como argumento da função também o **tamanho** do vetor. Sem ele, a função não tem como saber o tamanho do vetor que foi passado a ela como argumento.

O programa main :

```
main()
{
    int idade[5], i,soma;
    for(i = 0; i < 5; i++)
    {
        printf("Digite a idade da pessoa %d: ", i);
        scanf("%d", &idade[i]);
    }
    soma= media(idade, 5);
    printf("Idade media: %d\n",soma);
}
```

Na função main(), chamamos a função media() passando dois atributos: idade e 5.

Passagem de parâmetros do tipo vetor e sempre feita por *referência*.

Dessa forma, **qualquer ação realizada no vetor afetara o conteúdo do vetor passado como argumento**; ou seja, a

Exemplo: No programa abaixo, a função dobro recebe como parâmetro dados[] e multiplica por dois cada um dos valores do vetor na função

```

#include <stdio.h>
#include <conio.h>

void dobra(int numero, int dados[])
{
    int i;
    for (i=0; i<numero; i++)
        dados[i]= dados[ i ]*2;
}

main()
{
    int v[10], i;
    for(i=0;i<10;i++)
    { printf("entre com o elemeno %i do vetor = ",i);
      scanf("%i",&v[i]);
    }
    sdobra(10,v);
    for(i=0;i<10;i++)
        printf( " o elemeno %i do vetor = %i\n",i,v[i]);

    getch();
}

```

Inicialização

A linguagem C permite que vetores sejam inicializados. No caso, será inicializada uma variável contendo o número de dias de cada mês:

```
int numdias[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

O número de elementos do vetor pode ser omitido, ou seja,

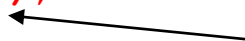
```
int numdias[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Matrizes como argumentos de funções

A passagem de uma matriz para uma função é similar à passagem de um vetor. O método de passagem do endereço da matriz para a função é idêntico, não importando quantas dimensões ela possua, já que **sempre é passado o endereço da matriz**.

Entretanto, na declaração da função, a matriz é um pouco diferente. A função deve receber o tamanho das dimensões a partir da segunda dimensão. Por exemplo:

```
void Determinante(float A[][5]);
```



Note que é fornecido a segunda dimensão da matriz. Isto é necessário para que, ao chamarmos o elemento `A[i][j]`, a função saiba a partir de que elemento ela deve mudar de linha.

Classe de Variáveis

Variáveis locais

As variáveis que são declaradas dentro de uma função são **chamadas de locais**. Na realidade toda variável declarada entre um bloco `{ }` podem ser referenciadas apenas dentro deste bloco. Elas existem apenas durante a execução do bloco de código no qual estão declaradas.

Variáveis Globais

São conhecidas por todo programa e podem ser usadas em qualquer parte do código. Permanecem com seu valor durante toda execução do programa. Deve ser **declarada fora de qualquer função** e até mesmo antes da declaração da função `main`. Fica numa região fixa da memória própria para esse fim.

Comando define

Pode-se definir uma variável global com a dimensão máxima da matriz.

```
#include<stdio.h>
#include<conio.h>
```

```
#define n 20
```

“n” é uma variável global.
Está definido fora da **função**
e do **main**

```
int func(int a[ ][n], int m, int l )
{
    int i, j;
    int cont=0;
    for(i=0;i<m;i++)
        for(j=0;j<l;j++)
            .....
            .....
            .....
}
```

```

int main()
{
    int m, i, j, num, kont;

    int mat[n][n];

    printf("Digite o numero de linhas da matriz  \n");
    scanf("%d",&linhas);

    printf("Digite o numero de linhas da matriz  \n");
    scanf("%d",&colunas);

    for(i=0;i<linhas;i++)
    {
        for(j=0;j<colunas;j++){
            printf("Digite termo %d,%d da matriz\n",
i,j);
            scanf("%d", &mat[i][j]);
        }

        func(mat, linhas,colunas);
        .....
        .....
    }
}

```

Deve-se alocar o tamanho máximo da matriz mesmo que depois o usuário entre com o tamanho real

```

#include <stdio.h>
#include <conio.h>

#define LINHAS 3
#define COLUNAS 3

void main ()
{
    int matriz[LINHAS][COLUNAS];
    int vetor[LINHAS];

    Le_Matriz (matriz);

    Imprime_Matriz (matriz);

    Soma_Linhas (matriz, vetor);

    Imprime_Vetor (vetor);
    getch();
}

```

Inicialização

As matrizes são inicializadas como os vetores, ou seja, os elementos são colocados entre chaves e separados por vírgulas. Como seus elementos são vetores, estes, por sua vez, também são inicializados com seus elementos entre chaves e separados por vírgulas. Por exemplo:

```
int A[5][6];
```

```
int A[5][6] = { { 1, 2, 3, 4, 5, 6},  
                { 7, 8, 9, 10, 11, 12},  
                {13, 14, 15, 16, 17, 18},  
                {19, 20, 21, 22, 23, 24},  
                {25, 26, 27, 28, 29, 30} };
```