

INF01202

Algoritmos e Programação

Modalidade Ead – Turma H

**Material de apoio: capacitar a programação
usando funções com parâmetros tipo ponteiro**

Tipos de passagem de parâmetros para uma função:

Por valor: apenas a cópia dos valores dos parâmetros são passados para a função. Durante a execução da função, alterações sobre esses parâmetros podem ser realizadas mas, concluída a execução da função, são perdidas. E no programa que chamou a função, os valores originais utilizados na sua ativação permanecem inalterados.

Por referência: essa modalidade de passagem de parâmetros permite que alterações realizadas sobre um parâmetro possam ser percebidas também no programa que chamou a função, tanto durante a execução da função, como após o seu término.

Em C, NÃO EXISTE passagem de parâmetros por referência.

Como o C contorna não ter passagem de parâmetros por referência

Sem passagem de parâmetros por referência, é impossível tornar visível fora das funções as alterações ocorridas no seu interior, a menos que essas alterações sejam apresentadas enquanto a função está ativa.

Não tendo passagem de parâmetros por referência, o C trabalha com a passagem de parâmetros com endereços, ou seja, ponteiros, para ter acesso aos resultados das funções após encerrada sua execução.

Exemplo 1: passagem por valor

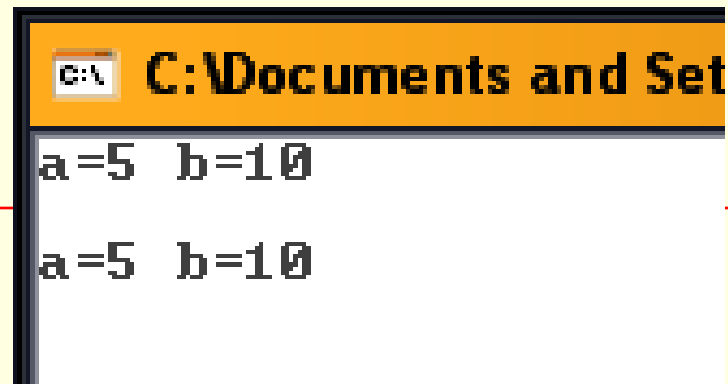
- Fazer uma função *void* que receba 2 inteiros x e y e troque seus valores.

```
#include <stdio.h>
#include <stdlib.h>
void troca (int, int);
int main( )
{

    int a=5, b=10;
    printf("a=%d b=%d\n",a,b);
    troca(a,b);
    printf("a=%d b=%d\n",a,b);
    system("pause");
    return 0;
}
```

```
void troca(int x, int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

O que será impresso na tela?



```
C:\> C:\Documents and Settings\...>
a=5 b=10
a=5 b=10
```

- Os valores não foram trocados porque a passagem de **parâmetros** foi feita por **valor**.
- Dentro do subprograma, os valores de x e y foram trocados, mas esta mudança foi feita sobre as **cópias dos valores** e não foi propagada para fora da sub-rotina.

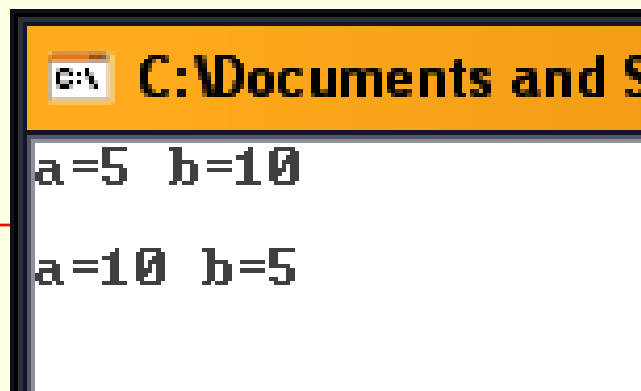
Exemplo 2: passagem usando ponteiros

- Fazer uma função *void* que receba 2 inteiros x e y e troque seus valores.

```
#include <stdio.h>
#include <stdlib.h>
void troca (int *, int *);
int main( )
{
    int a=5, b=10;
    printf("a=%d b=%d\n",a,b);
    troca(&a,&b);
    printf("a=%d b=%d\n",a,b);
    system("pause");
    return 0;
}
```

```
void troca(int *x, int *y)
{
    int temp;
    temp=*x;
    *x = *y;
    *y = temp;
}
```

x e y agora recebem os endereços de a e b



A screenshot of a Windows command prompt window. The title bar is orange and contains the text "C:\Documents and S...". The command prompt shows two lines of text: "a=5 b=10" on the first line and "a=10 b=5" on the second line. A horizontal red line is drawn across the slide, passing behind the command prompt window.

- Os valores foram trocados, pois a passagem de **parâmetros** foi feita usando **endereços**.
- No subprograma, x e y receberam os **endereços de memória** de a e b.

Relembrando a ligação de ponteiros com vetores e *strings*:

- Os ponteiros são normalmente utilizados na manipulação de vetores e *strings*.
- O nome de um vetor é um ponteiro que aponta para o **primeiro elemento** do vetor. Portanto,
 - Se *v* for um vetor Então *v* == &*v*[0]
- Considerando:
 - `int v[3] = {10, 20, 30};` //vetor *c*/ 3 inteiros
 - `int *ptr;` //ponteiro *p*/ inteiro
- Existem 2 formas p/ fazer com que *ptr* aponte p/ *v* :
 - `Ptr = &v[0];`
 - `Ptr = v;`
- O ponteiro *v* não deveria ser alterado durante a execução do programa, mas *ptr* PODE!

Parâmetros do tipo vetor (ou matriz)

Quando um vetor (ou matriz) é passado como argumento para uma função, não é o vetor (ou a matriz) na sua integralidade que é repassado, mas apenas o seu **endereço** inicial.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
void display(int num[MAX]);
int main( )
{
    int t[MAX], i;
    for(i=0; i<MAX; i++)
        t[i]=i;
    display(t);
    system("pause");
    return 0;
}
```

```
void display(int num[MAX])
{
    int i;
    for (i=0; i<MAX; i++)
        printf("%d\n", num[i]);
}
```

Parâmetros do tipo vetor (ou matriz): declaração como vetor sem dimensão

O parâmetro é especificado como um **vetor sem dimensão**.



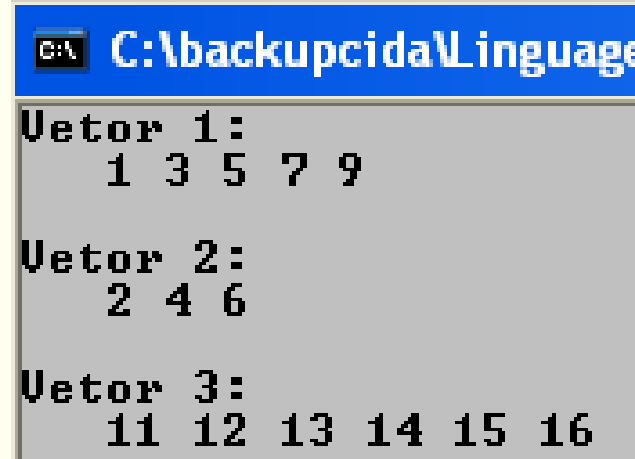
```
#include <stdio.h>
#include <stdlib.h>
#define MAX 3
void display(int num [ ], int);
int main( )
{
    int t[MAX], i;
    for(i=0; i<MAX; i++)
        t[i]=i;
    display(t, MAX);
    system("pause");
    return 0;
}
```

```
void display(int num[ ], int maxi)
{
    int i;
    for (i=0; i<maxi; i++)
        printf("%d\n", num[i]);
}
```

```

// Funcao void com parâmetro vetor e inteiro
#include <stdio.h>
#include <stdlib.h>
// impressão de vetores inteiros com numelem elementos
void impr_vetor(int num[ ],int numelem)
{
    int i;
    for (i=0; i < numelem; i++)
        printf("%d ",num[i]);
}
int main( )
{
    int vet1[5] = {1,3,5,7,9}, vet2[3] = {2,4,6};
    int vet3[6] = {11,12,13,14,15,16};
    system("color f1");
    printf("Vetor 1:\n  ");
    impr_vetor(vet1 , 5); // argumento: vetor de 5 elementos
    printf("\n\nVetor 2:\n  ");
    impr_vetor(vet2 , 3); // argumento: vetor de 3 elementos
    printf("\n\nVetor 3:\n  ");
    impr_vetor(vet3, 6); // argumento: vetor de 6 elementos
    printf ("\n\n");
    system("pause");
    return 0;
}

```



```

C:\backupcida\Language
Vetor 1:
  1 3 5 7 9

Vetor 2:
  2 4 6


Vetor 3:
 11 12 13 14 15 16

```

Parâmetro do tipo vetor (ou matriz): declaração como ponteiro

O parâmetro é especificado **como um ponteiro**.

```
#include <stdlib.h>
#include <stdio.h>
#define MAX 5
void display(int *, int);
int main( )
{
    int t[MAX], i;
    for(i=0;i<MAX;i++)
        t[i]=i;
    display(t, MAX);
    printf("\n\n");
    system("pause");
    return 0;
}
```



```
void display(int *num, int maxim)
{
    int i;
    for (i=0; i<maxim; i++)
        printf("%d ", num[i]);
}
```

Exemplos de programas usando funções *void* com parâmetros ponteiros

Exercício 1:

Preencher 3 vetores *v1*, *v2*, e *v3* com valores reais, valendo-se da função *rand*. Usando o procedimento *maior*, preencher o vetor *vmaior* com os valores dos maiores elementos de cada posição dos três vetores considerados. Imprimir todos os vetores, através da função *imprime_vetor*.

Funções utilizadas:

Função *imprime_vetor*: utiliza um parâmetro vetor. Para a função é repassado o endereço inicial desse vetor.

Função *maior*: utiliza três parâmetros por valor, que são os valores originários de cada um dos três vetores lidos, e um parâmetro ponteiro (endereço), que permite armazenar o maior dos três valores lidos em um vetor de maiores.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // funcao time
#define MAX 10
```

Com protótipos
das funções

```
void imprime_vetor(float [ ]); // vetor de reais
void maior (float, float, float, float *); // reais e ponteiro
//para real
```

```
int main( )
{
    float v1[10],v2[10],v3[10],vmaior[10];
    int i;
    system("color f1");
    // iniciando semente da funcao randomica
    srand(time(NULL));
    //criando os vetores:
    for (i=0; i<MAX; i++)
    { // gera numeros entre 1 e 10 e divide
        v1[i]= (rand( )%10 + 1)/ 0.23;
        v2[i]= (rand( )%10 + 1)/ 0.23;
        v3[i]= (rand( )%10 + 1)/ 0.23;
        maior(v1[i],v2[i],v3[i],&vmaior[i]); // valores float
    } ... // impressao dos vetores
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // funcao time
#define MAX 10
```

Com as funções
declaradas no início

```
void imprime_vetor(float vet[ ])
{ ... }
void maior (float x1, float x2, float x3, float *o_maior)
{ ... }
int main( )
{
    float v1[MAX ],v2[MAX ],v3[MAX ],vmaior;
    int i;
    system("color f1");
    // iniciando semente da funcao randomica
    srand(time(NULL));
    //criando os vetores:
    for (i=0; i<MAX; i++)
    { // gera numeros entre 1 e 10 e divide:
        v1[i]= (rand( )%10 + 1)/ 0.23;
        v2[i]= (rand( )%10 + 1)/ 0.23;
        v3[i]= (rand( )%10 + 1)/ 0.23;
        maior(v1[i],v2[i],v3[i],&vmaior[i]); // valores float
    } ... // impressao dos vetores
```

```
Vetor 1:
21.74 30.43 34.78 34.78 30.43

Vetor 2:
30.43 43.48 26.09 30.43 21.74

Vetor 3:
13.04 13.04 13.04 34.78 8.70

Vetor com os maiores:
30.43 43.48 34.78 34.78 30.43

Pressione qualquer tecla para continuar. . .
```



```

#include <stdio.h>
#include <stdlib.h>
#include <time.h> // funcao time
#define MAX 10
// imprime um vetor float de MAX elementos:
void imprime_vetor(float vet[])
{
    int i;
    for (i=0; i<MAX; i++)
        printf("%6.2f ",vet[i]);
    printf("\n");
}

```

Função imprime_vetor:

parâmetro vetor: float vet []

```

// recebe três valores reais e devolve o maior dos três
void maior (float x1, float x2, float x3, float *o_maior)
{
    *o_maior = x1;
    if (x2 > *o_maior)
        *o_maior = x2;
    if (x3 > *o_maior)
        *o_maior = x3;
}

```

Função maior: parâmetros reais: x1, x2, x3, parâmetro ponteiro para float: *o_maior.

```

int main( ) // slide anterior
{
    ...
}

```

```

int main( )
{
    float v1[MAX ],v2[MAX ],v3[MAX ],vmaior[MAX ];
    int i;
    system("color f1");
    // iniciando semente da funcao randomica:
    srand(time(NULL));
    //criando os vetores
    for (i=0; i<MAX; i++)
    { // gera nros entre 1 e 10 e divide:
        v1[i]= (rand( )%10 + 1)/ 0.23;
        v2[i]= (rand( )%10 + 1)/ 0.23;
        v3[i]= (rand( )%10 + 1)/ 0.23;
        maior(v1[i],v2[i],v3[i],&vmaior[i]);
    }
    // impressao dos vetores
    printf("\nVetor 1:\n");    imprime_vetor(v1);
    printf("\nVetor 2:\n");    imprime_vetor(v2);
    printf("\nVetor 3:\n");    imprime_vetor(v3);
    printf("\nVetor com os maiores:\n");
    imprime_vetor(vmaior);
    printf("\n");
    system("pause");
    return 0;
}

```

Exercício 2:

Preencher 2 vetores (20) de inteiros por leitura. Usando a função void **troca**, trocar os conteúdos desses 2 vetores

Função utilizada:

Função **troca**: recebe dois ponteiros (endereços) inteiros, apontando para uma posição em cada um dos vetores lidos.
Troca os valores dessas posições.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int main( )
```

```
{
    int vet1[MAX], vet2[MAX];
    int i;
    for (i=0;i<MAX;i++)
    {
        printf("digite o valor %d para os 2 vetores\n", i+1);
        scanf("%d%d", &vet1[i],&vet2[i]);
        troca(&vet1[i],&vet2[i]);
    }
    printf("imprimindo os vetores com os valores trocados\n");
    for (i=0;i<MAX;i++)
        printf("%5d %5d\n", vet1[i], vet2[i]);
    system("pause");
    return 0;
}
```

Função **troca**: parâmetros
ponteiros para inteiros: x e y.

```
void troca(int *x,int *y)
{
    int temp;
    temp=*x;
    *x = *y;
    *y = temp;
}
```