Técnicas de Programação II

Passagem de Parâmetros

Prof. José Luiz

Funções

- Uma função é composta por quatro partes:

1. Nome da função
2. Tipo de retorno
3. Lista de parâmetros
4. Corpo da função

Funções

Exemplo: função Maior

```
int Maior(int a, int b)
{
        return (a > b) ? a : b;
}
```

Nome da Função:	Maior
Tipo de Retorno	int
Lista de Parâmetros	int a, int b
Corpo da Função	return (a > b) ? a : b;

Uma função que não retorne qualquer tipo, isto é, que "retorne" void, chama-se vulgarmente procedimento. Mas não deixa por isso em C, de ser uma função

Funções

- Um exemplo mais significativo ocorre com a função **scanf**, que retorna um inteiro indicando o número de variáveis lidas com sucesso, embora normalmente ignoremos esse valor.

```
1. #include <stdio.h>
2. int main()
3. {
4.         int n
5.         puts(“Digite um nº inteiro: “);
6.         while (scanf(“%d”, &n) == 0)
7.                 fflush(stdin);
8.          printf(“Valor inteiro: %d\n”,n);
```

Passagem de parâmetros por valor

- Sempre que a passagem de parâmetros é realizada por valor não é a variável ou expressão que é enviada para a função, mas sim uma **cópia do seu valor**.

- Como consequência, dentro da função poderemos alterar como quisermos os valor que recebemos como parâmetros que o valor original das variáveis não sofrerá qualquer alteração, pois estamos alterando cópias dos valores originais.

- Uma vez terminada a função o programa continua executando os valores originais das variáveis invocadoras.

Passagem de parâmetros por referência

- Na passagem de parâmetros por referência o que é enviado para a função não é uma cópia do valor da variável, mas sim a própria variável ou uma referência a esta.

- Dessa forma, qualquer alteração nos parâmetros da função corresponde, na realidade, a uma alteração nas próprias variáveis referenciadas na invocação à função.

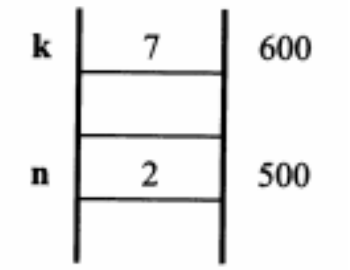
Passagem de parâmetros por referência

- Tomemos por exemplo a função troca:

  void troca (int *a, int *b);

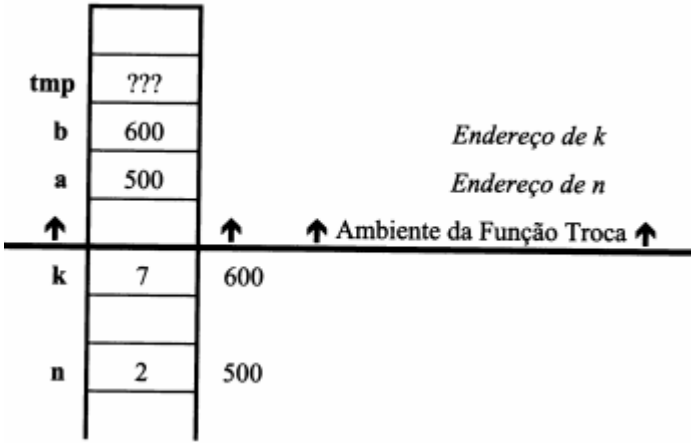
  com chamada da função: troca(&n, &k);

- A invocação da função troca será realizada com o endereço das variáveis, e não com o seu valor.
- Assim a função receberá como parâmetros, dois ponteiro para inteiros
- Suponhamos que os endereços das variáveis **n** e **k** eram, respectivamente, 500 e 600.



Passagem de parâmetros por referência

- Quando a função é executada ela estará usando ponteiros

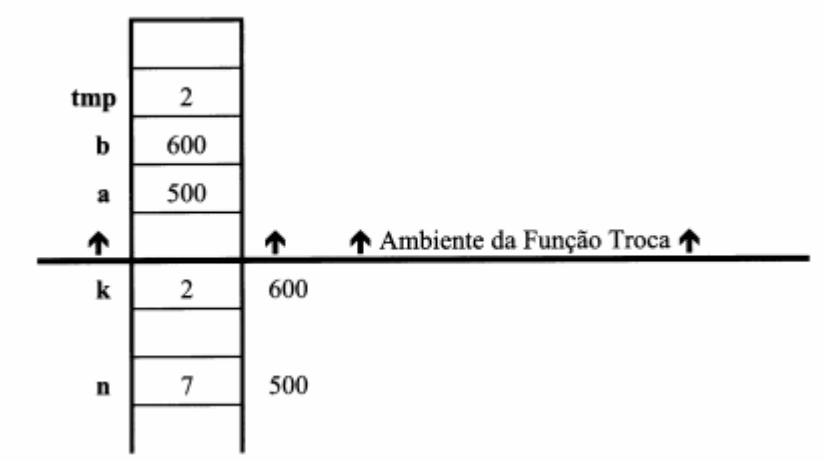


- Será necessário utilizar uma variável auxiliar para realizar a troca
- A troca passo a passo: armazenar um dos valor da troca (o de **n**)

```
tmp = *a;        // tmp recebe o conteúdo apontado por a = 2
*a = *b;         // coloca no endereço 500 o conteúdo do end. 600
*b = tmp;        // coloca em k o valor armazenado em tmp
```

Passagem de parâmetros por referência

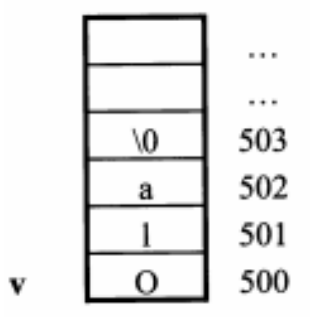
- Em seguida a função termina, sendo eliminada toda a memória utilizada para sua execução.



- Estando **n** e **k** com os valores trocados, como era pretendido.

Passagem de vetores para funções

- Como se pode observar, o nome de um vetor é, em si mesmo, um endereço. Por isso, sempre que passamos um vetor para uma função é enviado apenas o endereço original do vetor e não o vetor na sua totalidade.
- Suponhamos a seguinte declaração: **char v[ ] = “Ola”;**
- Que corresponde ao seguinte esquema:



Suponhamos que o endereço inicial do vetor é 500. Todos os elementos de um vetor ocupam posições consecutivas de memória.

Passagem de vetores para funções

- Exemplo: uma função para colocar um caracter **ch** em todas as posições da *string* **str**

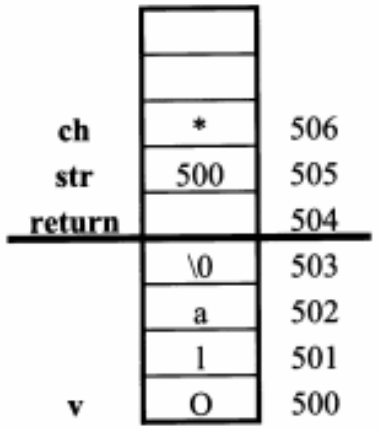
char* strset(char *str, char ch);

- para carregar a *string* **v** com asteriscos, podemos fazer:

strset (v, '*');

```
char* strset(char *str, char ch)
{
        int i;
        for(i=0; str[i] != '\0'; i++)
                str[i] = ch;
        return str;          // retorna o endereço inicial de str
}
```

Chamada da função: **strset(v, '*');**



Passagem de vetores para funções

- No caso de um vetor unidimensional precisamos colocar apenas o par de colchetes – **[ ]** – após o nome da variável, e vazio.
- Não importa se seu vetor tem 1, 10 ou 1 milhão de elementos, mas precisa também do número de elementos do vetor (exceto *strings*).
- As funções em C apenas precisam saber qual o tipo da variável e se é um vetor.
  Então, a sintaxe de uma função que recebe um vetor é:

  tipo_retorno nomeDaFuncao(tipo vetor[] , int tam);

- Por exemplo, uma função que recebe um vetor de inteiros e retorna um inteiro:

  int funcao(int numeros[], int tam)

Chamada da função: **funcao(numeros,10);**

Passagem de matriz para funções

- Sabemos que matrizes multidimensionais são aqueles que possuem duas ou mais dimensões.
- O detalhe para passar matriz para as funções é que o primeiro par de colchetes pode sempre ir vazio, e os demais preenchidos.
- Por exemplo, vamos supor que você criou um jogo da velha, que é uma matriz 3x3, e uma função que checa se algum jogador ganhou o jogo completando alguma linha.
- Essa função teria o seguinte cabeçalho:

  int checaLinha(int matriz[][3], int n)

- Ou seja, não precisamos especificar o número de linhas que a matriz tem, mas somente o número de colunas (mas precisamos informar também o número de linhas de maneira análoga a passagem de vetor).