Funções (parte 2)

Ponteiros (parte 1)

Prof. Debora Medeiros

Baseado no material de:

Ciro Trindade (Unisantos)

Passando Argumentos para Funções

- Argumento
  - mecanismo usado p/ transmitir informações a uma função
  - também é chamado de parâmetro
- O argumento é uma nova variável que armazena a informação passada p/ a função
  - Funciona exatamente como uma variável local
  - criada quando a função inicia sua execução
  - destruída quando a função termina

Passando Argumentos para Funções

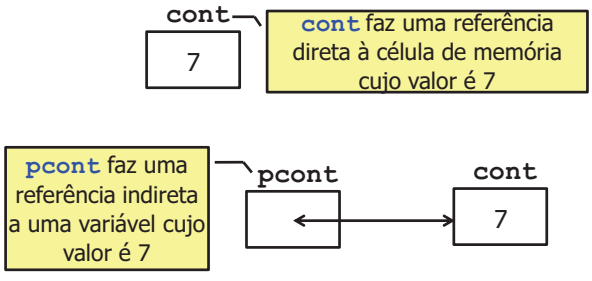
- Argumentos - chamada por valor
  - é dada uma cópia dos valores dos argumentos à função
  - ela cria variáveis temporárias para armazenar estes valores
  - A função chamada não altera o valor de uma variável de onde é chamada
    - ela só pode alterar sua cópia temporária
  - E se eu quiser alterar o valor de uma variável?

Ponteiros

Conceito de Ponteiro

- Variáveis especiais que armazenam endereços de memória
- Variável comum
  - **referência direta a um valor específico**
- Ponteiro
  - endereço de uma variável que contém um valor específico
  - **referência indireta ao valor da variável cujo endereço ele armazena**

Conceito de Ponteiro



Declarando Variáveis do Tipo Ponteiro

- Sintaxe:
  - `tipo * variável;`
- Exemplos:
  - `int * pcont;`
  - `float * px, * py;`
- Todo ponteiro tem um tipo específico que indica o tipo da variável para o qual ele aponta
  - Ex.: Um ponteiro inteiro deve armazenar o endereço de memória de variáveis inteiras

Operadores de Ponteiros

- 2 operadores unários são usados:
  - **&**: obtém o endereço de memória de uma variável
  - *: obtém o conteúdo do endereço de memória armazenado em um ponteiro
- Exemplo:

```
int cont = 7;
int * pcont = &cont;
printf("%d\n",*pcont);
```


Exemplo

```
#include<stdio.h>
int main() {
  int x = 10;// x é um inteiro
  int * px;  // px é um ponteiro p/ inteiro
  px = &x;  // px recebe o endereço de x
  printf("Endereço de x: %p\n", &x);
  printf("Valor de px: %p\n", px);
  printf("Valor de x: %d\n", x);
  printf("Valor do endereço armazenado em px: %d\n",
  *px);
  /* alterando dados */
  *px = 30;
  printf("Valor de x: %d\n", x);
  printf("Valor de px: %p\n", px);
  system("pause");
  return 0;
}
```


Exemplo

```
#include<stdio.h>
int main() {
  int x = 10;// x é um inteiro
  int * px;  // px é um ponteiro p/ inteiro
  px = &x;  // px recebe o endereço de x
  printf("Endereço de x: %p\n", &x);
  printf("Valor de px: %p\n", px);
  printf("Valor de x: %d\n", x);
  printf("Valor do endereço armazenado em px: %d\n",
  *px);
  /* alterando dados */
  *px = 30;
  printf("Valor de x: %d\n", x);
  printf("Valor de px: %p\n", px);
  system("pause");
  return 0;
}
```

```
Endereco de x: 0022FF44
Valor de px: 0022FF44
Valor de x: 10
Valor do endereco armazenado em px: 10
Valor de x: 30
Valor de px: 0022FF44
Press any key to continue . . .
```


Passagem de Parâmetros por Referência

- Permite à função alterar os valores das variáveis que foram passadas como parâmetro
- Na chamada à função, são transmitidos os endereços das variáveis
  - Usa-se o operador &
  - Os parâmetros da função devem ser ponteiros

Passagem de Parâmetros por Referência

- Exemplo
  - Função que troca os valores de variáveis
  - Pode ser usado em ordenação
  - ...

Exemplo

```c
#include <stdio.h>

void troca(int * pa, int * pb) {
    int aux = *pa;
    *pa = *pb;
    *pb = aux;
}

int main() {
    int a = 10,b = 20;

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


Exemplo

```c
#include <stdio.h>

void troca(int * pa, int * pb) {
    int aux = *pa;
    *pa = *pb;
    *pb = aux;
}

int main() {
    int a = 10,b = 20;

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

```
a = 10, b = 20
a = 20, b = 10
Press any key to continue . . .
```


Vetores e Matrizes como Argumentos de Funções

- Chamamos a função com o nome do vetor/matriz sem nenhum índice
- Passamos o endereço do 1º elemento do vetor/matriz para a função
- Exemplo:

```c
int main() {
  int vet[10];
  func1(vet);
  ...
}
```


Vetores e Matrizes como Argumentos de Funções

- Se a função recebe um vetor, o parâmetro formal pode ser um ponteiro, um vetor dimensionado, ou um vetor sem dimensão

```c
void fun(int *a)        void fun(int a[10])     void fun(int a[])
{                       {                       {
 ...                     ...                     ...
}                       }                       }
```

- Esses 3 métodos dizem ao compilador que um ponteiro para um inteiro está sendo recebido
- Passagem de parâmetros por referência

Vetores e Matrizes como Argumentos de Funções

- Se a função recebe uma matriz bidimensional como argumento, o número de elementos da segunda dimensão (colunas) deve ser incluído no cabeçalho da função
- Exemplo: suponha **a** uma matriz 4 x 6

```c
void fun(int a[4][6])      void fun(int a[][6])
{                          {
 ...                        ...
}                          }
```


- Programa de ordenação de strings
  - Função que troca duas strings

```c
#include <stdio.h>
#include <string.h>

void strtroca(char s1[],char s2[]){
     char aux[80];
     strcpy(aux,s1);
     strcpy(s1,s2);
     strcpy(s2,aux);
}

int main() {
    int MAX=100, LEN=80, i, j,troca=1;
    char texto[100][80];

    printf("Digite uma linha vazia para sair.\n");
    for(i = 0; i < MAX; i++) {
        printf("%03d: ",i+1);
        fgets(texto[i],LEN,stdin);
        if(texto[i][0] == '\n') {
            break;
        }
    }
    while(troca){
      troca=0;
      for(j=0;j<(i-1);j++){
        if (strcmp(texto[j],texto[j+1])>0){
           strtroca(texto[j],texto[j+1]);
           troca=1;
        }
      }
    }
    // imprime todo o conteúdo da matriz
    for(j = 0; j < i; j++) {
        printf("%03d: %s",j+1,texto[j]);
    }
    system("pause");
    return 0;
}
```

Exemplo

- Função que calcula a transposta de uma matriz
  - Quadro…