Funções em Linguagem C
Parte II

Profa. Dra. Elisa Yumi Nakagawa

1. Semestre 2017

Slides inicialmente preparados pelo Prof. Dr. Claudio Fabiano Motta Toledo

Sumário

- Introdução a Ponteiros
- Escopo de Variáveis
- Variáveis locais x Variáveis globais
- Passagem de parâmetros por referência

Ponteiros

- Uma **variável** é armazenada em um certo número de bytes em uma determinada posição de memória.
- Um **ponteiro** é uma variável que contém o endereço de outra variável.
- Os **ponteiros** são usados para acessar e manipular conteúdos em determinado endereço de memória.

Ponteiros

- Acesso ao endereço de memória da variável:

  &<nome_var>

- Declaração de um ponteiro:

  <tipo> *<nome_var_ponteiro>

- Exemplos de declaração de ponteiros:

```
int *p;
char *q;
float *r,*s;
```

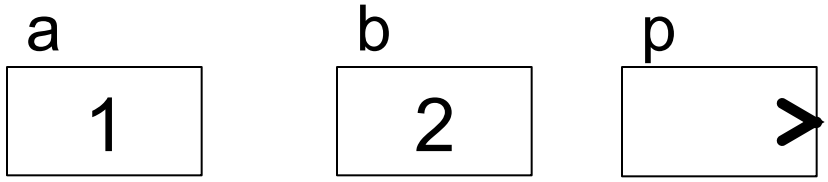
Ponteiros

- O operador unário * representa indireção ou deferenciação.
- Se p é um ponteiro, então *p é o valor da variável da qual p é o endereço.
- O valor direto de p é o endereço de memória.
- *p é o valor indireto de p, pois representa o valor armazenado no endereço de memória.

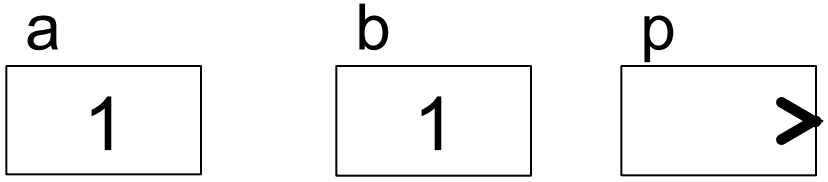
Ponteiros

int a=1, b=2, *p;

p=&a



b = *p;



Ponteiros

Exemplo:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i=7, *p = &i;
    printf("%s%d\n%s%p\n", "Valor de i: ", *p, "Endereco de i:", p);
    return 0;
}
```

```
Valor de i: 7
Endereco de i: 0028FF44
```

Escopo de Variáveis

- Estabelece onde uma variável poderá ser utilizada em um programa.
- A regra básica envolvendo escopo é que os identificadores são acessados apenas dentro do bloco em que foram declarados.
- Os identificadores não são conhecidos fora dos limites do bloco onde foram declarados.
- Programadores podem escolher utilizar um mesmo identificador em diferentes declarações.

Variáveis locais x Variáveis globais

- **Variáveis Locais**
  - Tem como escopo a função onde foi declarada.
  - Nomes e valores dessas variáveis têm uso restrito à função que declarou essas variáveis.

- **Variáveis Globais**
  - Nomes e valores dessas variáveis podem ser acessados em todo o programa principal.
  - Essas variáveis devem ser declaradas fora do corpo de todos os procedimentos ou funções do programa.
  - O programa pode alterar uma variável global em qualquer ponto, tornando difícil localizar a alteração que possa ter ocasionado erro.

- **Exemplo**:

```
#include <stdio.h>
int a=1, b=2, c=3;            /* variáveis globais */
int f(void);                  /* protótipo da função*/
int main(void)
{
  printf("%3d\n", f( ));            / 12 é exibido */
  printf("%3d%3d%3d\n", a,b,c);   / 4 2 3 são exibidos*/
  return 0;
}

inf f(void)
{
  int b, c;       /* b e c são variáveis locais */
  a=b=c=4;
  return (a+b+c);
}
```

Passagem de parâmetros por referência em funções

- O endereço de memória da variável é fornecido à função e não uma cópia do valor da variável.
- Qualquer alteração executada pela função ocorre na posição de memória fornecida.
- Portanto, as alterações permanecem quando a função é encerrada.
- Ponteiros são usados nas passagens por referência.

- **Exemplo**:

```
#include <stdio.h>
void swap(int *, int *);
int main(void){
   int i=3, j=5;
   swap(&i, &j);
   printf("%d   %d\n", i, j);  /* 5  e 3 são exibidos */
   return 0;
}
void swap (int *p, int *q){
   int tmp;
   tmp = *p;
   *p = *q;
   *q = tmp;
}
```

Passagem de parâmetros por referência em funções

- As variáveis i e j são passadas por referência, ou seja, o endereço de memória das variáveis é repassado à função.

  swap (&i, &j)

- Os ponteiros *p e *q, declarados no argumento na função swap, passam a referenciar a posição de memória das variáveis i e j.

  void swap (int *p, int *q)

Passagem de parâmetros por referência em funções

- **Vetores e matrizes**
  - São passados sempre por referência.
  - Os colchetes, após o nome do vetor passado a uma função, indicam que o parâmetro é um vetor. Não precisa especificar o tamanho.

```
int soma_vetor(int vetor[], int elementos)
```

Passagem de parâmetros por referência em funções

- **Vetores e matrizes**

  - Exemplo: Formas possíveis de se declarar a função func() para receber o vetor int vet[100].

```
func(int x[100]);
func (int x[]);
func(int *x);
```

Passagem de parâmetros por referência em funções

- **Vetores e matrizes**
  - Ao passar uma matriz bidimensional, se for preciso acessar entradas específicas, o número de colunas precisa ser fornecido.

    int soma_matriz(int matriz[][5], int linhas)

  - Se não há necessidade de acessar entradas específicas da matriz, ela poderá ser tratada como um vetor. Nesse caso, o número de elementos da matriz deverá ser fornecido.

```c
#include<stdio.h>
#define LN  2
#define CL  5

int soma_vetor(int[], int);
int soma_matriz(int , int colunas,int matriz[][colunas]);

void main(){
    int vetor[CL]={10, 20, 30, 40, 50};

    int bidim[LN][CL]={{10,20,30,40,50},
                       {60,70,80,90,100} };
    printf("Soma dos valores do vetor:%d\n", soma_vetor(vetor,CL));
    printf("Soma dos valores da matriz:%d\n\n", soma_matriz(LN,CL,bidim));
    printf("Soma dos valores do matriz:%d\n", soma_vetor(bidim,LN*CL));
}

int soma_vetor(int vetor[], int elementos){
    int i,soma;

    soma=0;
    for(i=0; i<elementos; i++)
     soma += vetor[i];

    return soma;
}

int soma_matriz(int linhas, int colunas,int matriz[][colunas]){
    int i,j,soma;

    soma=0;
    for(i=0; i<linhas; i++)
     for(j=0; j<colunas; j++)
       soma += matriz[i][j];

     return soma;
}
```

```
Soma dos valores do vetor:150
Soma dos valores da matriz:550

Soma dos valores do matriz:550
```

Exercícios

1. Escreva um programa que receba um número inteiro representando a quantidade total de segundos e, usando passagem de parâmetros por referência, converta a quantidade informada de segundos em Horas, Minutos e Segundos. Imprima o resultado da conversão no formato HH:MM:SS. Utilize o seguinte protótipo de função:

void converteHora(int total_segundos, int *hora, int *min, int *seg);

2. Escreva um programa que leia um inteiro positivo n e uma sequência de n inteiros não-negativos, imprimindo o mdc de todos os números da sequência.

3. Escreva um programa que:

a) Recebe valores para as variáveis x1, x2 e x3. Os valores devem ser inteiros positivos. Crie uma função que valide a entrada de dados, o usuário só consegue entrar com valores adequados.

b) Devolva aos programa principal os valores ordenados com x1<x2<x3. Crie uma função que ordene esses valores.