Funções

Ponteiros (parte 1)

Prof. Debora Medeiros

Baseado no material de:
Ciro Trindade (Unisantos)

Por quê usar funções?

- Evitar a repetição de um conjunto de instruções que ocorre várias vezes no programa
- Modularizar o programa
- Reaproveitamento de código

Estruturas das Funções em C

```
tipo nome-da-função([lista-de-parâmetros])
  {
  corpo da função
  }
```

- tipo: especifica o tipo do valor que a função retorna
- lista-de-parâmetros: uma lista separada por vírgulas das variáveis que recebem os valores dos argumentos passados quando a função é chamada
- corpo da função: comandos delimitados por chaves { }

Exemplo de função

- Imprimir uma linha
- Ex:

```
------------------------------------------------
Calculadora
------------------------------------------------

Digite a operação desejada:
```


Exemplo

```c
#include <stdio.h>
void linha(); //protótipo da função linha()

int main() {
   linha();
   printf("\nUm programa em C\n");
   linha();
   system("pause");
   return 0;
}

void linha() {
   int i;
   for(i = 1; i <= 80; i++)
      printf("-");
}
```


Exemplo

```c
#include <stdio.h>
void linha(); //protótipo da função linha()

int main() {
   linha();
   printf("\nUm programa em C\n");
   linha();
   system("pause");
   return 0;
}

void linha() {
   int i;
   for(i = 1; i <= 80; i++)
      printf("-");
}
```

```
--------------------------------------------------------------------------------
Um programa em C
--------------------------------------------------------------------------------
Press any key to continue . . .
```

Chamando Funções

- Do mesmo modo que chamamos uma função da biblioteca C (`printf()`, `scanf()`, etc.) chamamos nossas próprias funções como `linha()`
- Os parênteses que seguem o nome são necessários para que o compilador possa diferenciar a chamada a uma função de uma variável

Variáveis Locais

- Variáveis declaradas dentro de uma função
  - conhecidas somente dentro do seu próprio bloco
  - Um bloco começa quando o compilador encontra uma chave de abertura ({) e termina quando o compilador encontra uma chave de fechamento (})
  - Variáveis locais existem apenas durante a execução do bloco de instruções onde estão declaradas

Funções que Devolvem um Valor

- O comando **`return`**
  - Causa uma saída imediata da função onde ele está contido
    - ou seja, termina a função
  - Pode ser usado para devolver um valor
  - Todas as funções, exceto aquelas do tipo void, devolvem um valor
  - valor explicitamente especificado pelo comando **`return`**
  - O comando **`return`** faz com que a chamada à função que contém o **`return`** seja substituída pelo valor que sucede este comando

Funções que Devolvem um Valor

- Desde que uma função não seja declarada como **`void`**, ela pode ser usada como um operando em qualquer expressão

```
if (max(x,y) > 100) printf("Maior");
printf("Valor absoluto de %d = %d", x, abs(x));
```

- Entretanto, uma função não pode receber uma atribuição

```
sqrt(x)=100;   // instrução errada
```


Exemplos

- Maior número (exercício das médias da lista)
  - max(x,y)

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

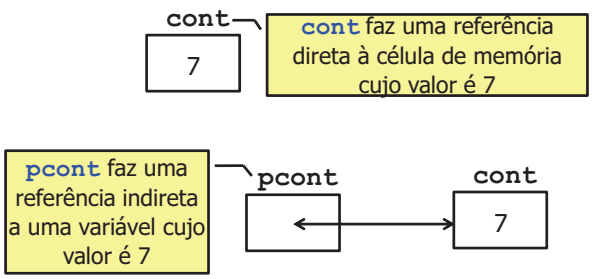
- Argumentos - chamada por valor
  - é dada uma cópia dos valores dos argumentos à função
  - ela cria variáveis temporárias para armazenar estes valores
  - A função chamada não altera o valor de uma variável de onde é chamada
    - ela só pode alterar sua cópia temporária
  - E se eu quiser alterar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

```c
#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

```c
#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 . . .
```