



Programação de Computadores I

Funções na Linguagem C

PROFESSORA CINTIA CAETANO

Introdução

- ▶ Funções são as estruturas que permitem ao usuário separar seus programas em blocos de construção.
- ▶ A principal característica das funções é o fato de, uma vez escritas e depuradas (testadas), elas poderem ser reutilizadas quantas vezes forem necessárias, inclusive por outros programas.

Sintaxe

<tipo de retorno> <nome da função> (declaração de parâmetros)

{ //início da função

...

...

...

} //final da função

Sintaxe

- ▶ O **<tipo de retorno>** é o tipo de variável que a função vai retornar. O default é o tipo int, ou seja, uma função para a qual não se declara o tipo de retorno é considerada como retornando um inteiro.
- ▶ A **<declaração de parâmetros>** é uma lista de nomes de variáveis separadas por vírgulas que recebem os valores dos argumentos quando a função é chamada. Ela possui a seguinte forma geral:
tipo nome_1, tipo nome_2, ... , tipo nome_n
- ▶ O **corpo da função** contém a declaração das variáveis locais, bem como a seqüência de comandos que serão executados, ou seja, o bloco de comandos da função.

Declaração de Parâmetros

- ▶ É na declaração de parâmetros que informamos ao compilador quais serão as entradas da função (assim como informamos a saída no `tipo_retorno`).
- ▶ O tipo deve ser especificado para cada uma das N variáveis de entrada.
- ▶ Cada variável descrita na declaração de parâmetros será tratada como uma variável local da função.
- ▶ Quando uma função não tiver argumentos de entrada, a lista de parâmetros será vazia. Entretanto, os parênteses da declaração da função são obrigatórios.

Retorno da Função

- ▶ Quando se encontra um comando **return**, a função é encerrada imediatamente e, se algum dado é informado após o return, seu valor é retornado pela função.
- ▶ É importante lembrar que, o valor de retorno fornecido tem que ser compatível com o tipo de retorno declarado para a função.
- ▶ Uma função pode ter mais de um comando return, entretanto, somente um será executado por chamada da função.
- ▶ Isto se torna claro quando pensamos que a função é terminada quando o programa chega ao primeiro comando return.

Exemplo 1

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

```
int quadrado(int a);
```

```
void main()
```

```
{
```

```
    int num;
```

```
    printf ("Entre com um numero: ");
```

```
    scanf ("%d",&num);
```

```
    num=quadrado(num);
```

```
    printf ("\n\n O quadrado é %d\n",num);
```

```
}
```

```
int quadrado(int a)
```

```
{
```

```
    return (a*a);
```

```
}
```

Exemplo 2

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

```
int EPar (int a) ;
```

```
int main()
```

```
{
```

```
    int num;
```

```
    printf ("Entre com numero: ");
```

```
    scanf ("%d",&num);
```

```
    if (EPar(num))
```

```
        printf ("\n\nO numero e par.\n");
```

```
    else
```

```
        printf ("\n\nO numero e impar.\n");
```

```
    return 0;
```

```
}
```

```
int EPar(int a)
```

```
{
```

```
    if (a%2) /* Verifica se a e divisivel por dois */  
        return 0; /* Retorna 0 se nao for divisivel */
```

```
    else
```

```
        return 1; /* Retorna 1 se for divisivel */
```

```
}
```

Exemplo 3

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
```

```
void area_quadrado()
{
    float area, a;
    printf("Entre com o lado do quadrado: ");
    scanf("%f",&a);
    area=pow(a,2); //area quadrado
    printf("A área do quadrado é: %4.2f",area);
}
```

```
void main()
{
    area_quadrado(); // chamada da função
}
```

Uso do Tipo Void em Funções

- ▶ Em inglês, void quer dizer vazio e é este o sentido deste tipo de dado.
- ▶ Ele permite a construção de funções que não retornam nada (procedimentos) e/ou de funções que não têm argumentos de entrada.
- ▶ A seguir são apresentados alguns exemplos de protótipos de funções que não retornam nada e/ou não possuem argumentos formais:

void nome_da_função (declaração_parâmetros); / Sem retorno */*

tipo_retorno nome_da_função (void); / Sem argumentos de entrada */*

void nome_da_função (void); / Sem argumentos formais e retorno */*

Uso do Tipo Void em Funções

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

```
void Mensagem (void);
```

```
void main (void)
```

```
{  
    Mensagem();  
    printf ("\tDiga de novo:\n");  
    Mensagem();  
}
```

```
void Mensagem ()
```

```
{  
    printf ("Ola! Eu estou vivo.\n");  
}
```

Chamada de Função

- ▶ Uma vez que as funções estejam definidas, pode-se usa-las sem se preocupar como elas foram escritas. Isto é o que denominamos chamada de função.
- ▶ A sintaxe geral da chamada de uma função é:
`nome_função(lista_valores);`
- ▶ Sendo a `lista_valores` uma lista com os valores que serão atribuídos a cada um dos argumentos formais declarados na função.
- ▶ OBS: a quantidade e o tipo dos valores declarados na `lista_valores` da chamada da função devem ser compatíveis com aqueles apresentados na `declaração_parâmetros` da declaração da função.

Chamada de Função

- ▶ Em geral, os argumentos podem ser passados para as funções de duas maneiras, conforme segue:
 1. Passagem de parâmetros por valor
 2. Passagem de parâmetros por referência

Passagem de parâmetros por valor

- ▶ Ou simplesmente “chamada por valor”.
- ▶ Os valores dos argumentos que são passados para a função são copiados nos parâmetros correspondentes.
- ▶ Assim, quaisquer alterações feitas nestes parâmetros formais não têm efeito nas variáveis empregadas na chamada da função.

Exemplo 4

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

```
float media(int a, int b);
```

```
void main ()
```

```
{
```

```
    int x, y;
```

```
    float sq;
```

```
    printf ("Entre com um numero: ");
```

```
    scanf ("%d",&x);
```

```
    printf ("Entre com outro numero: ");
```

```
    scanf ("%d",&y);
```

```
    sq = media(x,y);
```

```
    printf ("\n\n x = %d / y = %d\n",x,y);
```

```
    printf ("A média é: %4.2f\n",sq);
```

```
}
```

```
float media(int a, int b)
```

```
{
```

```
    float med;
```

```
    med = (a+b)/2;
```

```
    return med;
```

```
}
```

Passagem de parâmetros por referência

- ▶ Este tipo de chamada de função tem o nome de “chamada por referência”.
- ▶ Este nome vem do fato de que, neste tipo de chamada, não se passa para a função os valores das variáveis, mas sim suas referências(endereço da variável na memória).
- ▶ A função usa estas referências para alterar os valores das variáveis empregadas na chamada da função.

Passagem de parâmetros por referência

- ▶ Neste tipo de chamada, os parâmetros formais de uma função devem ser declarados como sendo ponteiros(utilizando o operador *).
- ▶ Os ponteiros são a “referência” que precisamos para poder alterar a variável fora da função.
- ▶ Para que estes ponteiros recebam o endereço de memória das variáveis utilizadas na chamada da função, é necessário colocar o operador & na frente de cada uma das variáveis, como apresentada no exemplo:

Exemplo 5

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

```
void sqr(float *num);
```

```
void main ()
```

```
{
```

```
    float nro;
```

```
    printf ("Entre com um numero: ");
```

```
    scanf ("%f",&nro);
```

```
    printf ("\n\nO numero original e: %f\n",nro);
```

```
    sqr(&nro);
```

```
    printf ("O seu quadrado vale: %f\n",nro);
```

```
}
```

```
void sqr (float *num)
```

```
{
```

```
    *num = (*num)*(*num);
```

```
}
```