Funções em C

Disciplina: PROGRAMAÇÃO II

Prof. Jean Eduardo Glazar
Curso de Sistemas de Informação
Campus Colatina-ES

INSTITUTO FEDERAL
Espírito Santo

Declaração de Funções

```
<tipo_retorno> <nome_função> ( <parâmetros> )
{
      <sequência de comandos> ;
       return <valor ou variável> ;
}
```

Onde:

<parâmetros> → <tipo_variável> <variável> ,
<tipo_variável> <variável> , . . .

Funções – Exemplo 1

```
int quadrado ( int x )
{
        int res;
        res = x * x;
        return res;
}
```

Devem ser do mesmo tipo.

Funções – Exemplo 2

```
float calcMedia( float n1 , float n2 , float n3 )
{
        float media;
        media = ( n1 + n2 + n3 ) / 3;
        return media;
}
```

Local de Implementação

- Em C as funções devem ser declaradas antes da função **main**, obedecendo o seguinte formato:

<Declaração de funções e procedimentos>

```
int main()
{
    <sequencia de comandos>
}
```

Chamada da Função

- Para utilizar uma função em C é bastante simples, basta utilizar a seguinte sintaxe:

<variável> = <nome_função> (<parâmetros de entrada>) ;

Chamada da Função – Exemplo 1

```
int quadrado ( int x ) {
        int res;
        res = x * x;
        return res;
}

int main() {
        int resQuad, num;
        // Ler o número num
        resQuad = quadrado (num);
}
```

Chamada da Função – Exemplo 2

```
float calcMedia( float n1 , float n2 , float n3 ) {
        float media;
        media = ( n1 + n2 + n3 ) / 3;
        return media;
}
int main() {
        float nota1, nota2, nota3, resMedia;
        // Ler as 3 notas
        resMedia = calcMedia (nota1 , nota2 , nota3);
}
```

Procedimentos

- Procedimento é uma função que não retorna nada.
- A linguagem C não trabalha o conceito de procedimentos, mas existe o tipo de retorno **void**, que significa que nada será retornado. Uma função que retorna **void** pode ser declarada da seguinte forma:

```
void <nome_função> ( <parâmetros> )
{
        <sequência de comandos> ;
}
```

Procedimentos – Exemplo

```
void menu ( ) {
        printf ("1 – Inserir\n");
        printf ("2 – Pesquisar\n");
        printf ("3 – Listar\n");
}

int main() {
        menu();
        . . .
}
```

Passagem de parâmetros por valor

- A forma de passagem de parâmetros padrão é a por valor, que é o tipo de passagem de parâmetros que NÃO permite a alteração de variáveis no interior de funções.

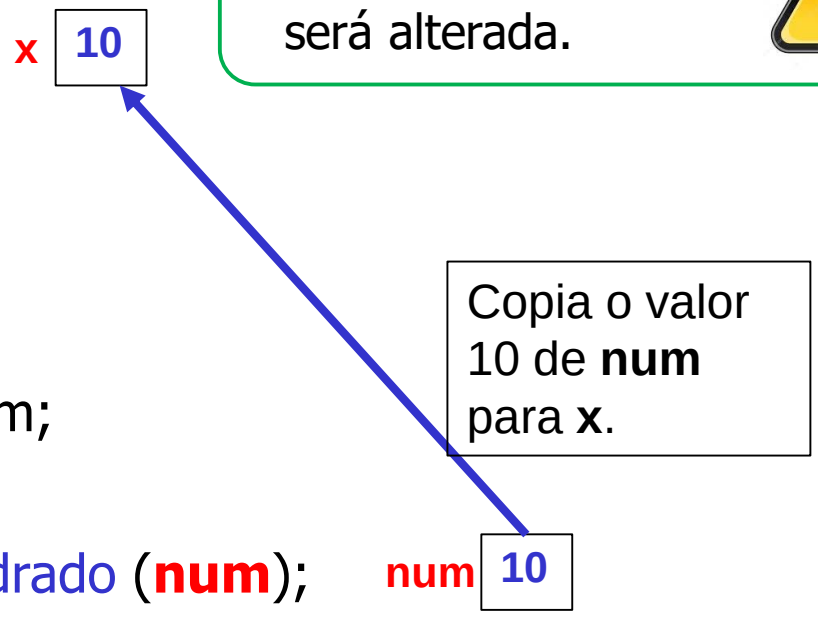
- O valor é copiado da variável da função principal para a variável do parâmetro da função.

Passagem de parâmetros por valor

```
int quadrado ( int x ) {
        int res;
        res = x * x;
        return res;
}

int main() {
        int resQuad, num;
        num = 10;
        resQuad = quadrado (num);
}
```

Se alterar a variável **x**, a variável **num** não será alterada.



Parâmetros por referência

- Consiste na possibilidade de alteração de valores de variáveis passadas como parâmetro para funções ou procedimentos.
- Em C não existe passagem de parâmetros por referência, para utilizar esse recurso são utilizados “ponteiros”.
- Nesse momento vamos entender o ponteiro apenas como uma forma de simular a passagem de parâmetros por referência

Parâmetros por referência

- Coloca-se um * antes da variável que se quer alterar.
- Na declaração do parâmetro:

<parâmetros> ➔ <tipo_variável> *<variável> ,
<tipo_variável> *<variável> , . . .

- Toda vez que usar a variável dentro da função:

***<variável> = . . .**

Chama-se essa variável de ponteiro.

Parâmetros por referência - Exemplo

```
void trocar ( int *n1 , int *n2 ) {
        int aux;
        aux = *n1;
        *n1 = *n2;
        *n2 = aux;
}
```

Parâmetros por referência

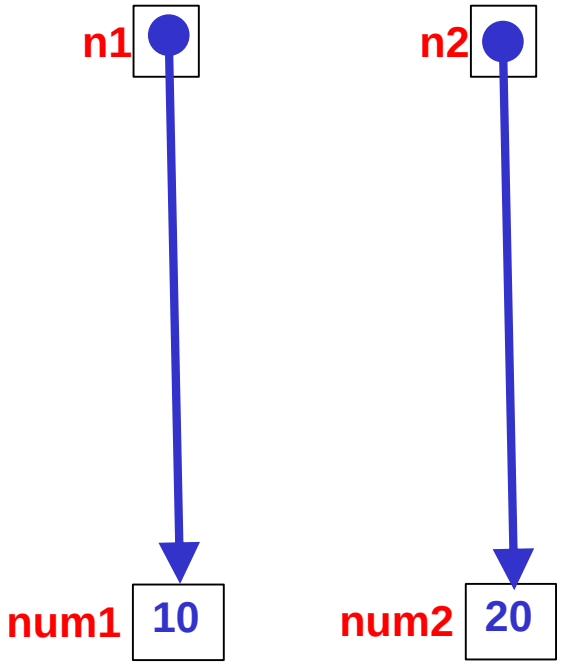
- Na chamada da função, coloca-se **&** antes da variável.

```
trocar(&num1, &num2);
```

Por isso que colocamos o **&** na função scanf, porque a variável será alterada.

Parâmetros por referência - Exemplo

```
void trocar ( int *n1 , int *n2 ) {
        int aux;
        aux = *n1;
        *n1 = *n2;
        *n2 = aux;
}
int main() {
        int num1, num2;
        num1 = 10;
        num2 = 20;
        trocar (&num1 , &num2 );
}
```



scanf com variável PONTEIRO

```
void lerDados( float *peso , float *altura ) {
        printf("Peso: ");
        scanf("%f", &*peso);     ➔ scanf("%f", peso);
        printf("Altura: ");
        scanf("%f", &*altura);   ➔ scanf("%f", altura);
}
```

Os símbolos **&** e * são opostos, então um anula o outro. Nesse caso, podemos usar sem nenhum símbolo.

scanf com variável PONTEIRO

```
void lerDados( float *peso , float *altura ) {
        printf("Peso: ");
        scanf("%f", peso);
        printf("Altura: ");
        scanf("%f", altura);
}
int main() {
        float p, a;
        lerDados(&p, &a);
        printf("Peso: %.1f\tAltura: %.1f\n", p, a);
}
```

INSTITUTO
FEDERAL
Espírito Santo