

# Funções

José Ricardo da Silva Junior

Adaptado de slides da prof. Profa. Maria Aparecida C.

# Principal Objetivo:

Facilitar a solução de problemas complexos.

“A arte de programar consiste na arte de organizar e dominar a complexidade dos sistemas”

Dijkstra, 1972

# Estratégia de Dividir para Conquistar

Divisão de um problema original em subproblemas (módulos) mais fáceis de resolver e transformáveis em trechos mais simples, com poucos comandos (subprogramas).

# Funções

- **Trechos de código independentes**, com estrutura semelhante àquela de programas, mas executados somente quando chamados por outro(s) trecho(s) de código.
- Devem executar UMA tarefa específica, muito bem identificada (**conforme a programação estruturada**).
- Ao ser ativado um subprograma, o fluxo de execução desloca-se do fluxo principal para o subprograma. Concluída a execução do subprograma, o fluxo de execução retorna ao ponto imediatamente após onde ocorreu a chamada do subprograma.

# Vantagens

- ◆ **Maior controle sobre a complexidade.**
- ◆ **Estrutura lógica mais clara.**
- ◆ **Maior facilidade de depuração e teste, já que subprogramas podem ser testados separadamente.**
- ◆ **Possibilidade de reutilização de código.**

# Funções em C

Funções são segmentos de programa que executam uma determinada tarefa específica.

Funções (também chamadas de rotinas, ou sub-programas) são a essência da **programação estruturada**.

Ex: `sqrt()`, `strlen()`, etc.

O programador também pode escrever suas próprias funções, chamadas de **funções de usuário**, que tem uma estrutura muito semelhante a um programa.

# Sintaxe

```
tipo_da_funcao  nome_da_função (lista_de_parâmetros)
{
    //declarações locais
    //comandos
}
```

**tipo\_da\_funcao:** o tipo de valor retornado pela função. Se não especificado, por falta a função é considerada como retornando um inteiro.

**nome\_da\_função:** nome da função conforme as regras do C

**lista\_de\_parâmetros:** tipo de cada parâmetro seguido de seu identificador, com vírgulas entre cada parâmetro. Mesmo se nenhum parâmetro for utilizado, os parênteses são obrigatórios.

Os parâmetros da declaração da função são chamados de parâmetros formais.

# Sintaxe

Ex.:

```
int soma_valores (int valor1, int valor2)
void imprime_linhas(int num_lin)
void apresenta_menu ( )
float conv_dolar_para_reais(float dolar);
```

# Sintaxe

//Escrita de numeros inteiros

```
#include <stdio.h>
#include <stdlib.h>
int main( )
```

```
{
```

```
    int i;
```

//apresentacao do cabecalho

```
    for (i=1;i<20;i++)
        printf("*");
    printf("\n");
```

```
    printf("Numeros entre 1 e 5\n");
```

```
    for (i=1;i<20;i++)
        printf("*");
    printf("\n");
```

//escrita dos numeros

```
    for (i=1;i<=5;i++)
        printf("%d\n",i);
```

```
    for (i=1;i<20;i++)
        printf("*");
    printf("\n");
```

```
    system("pause");
    return 0;
```

```
}
```

Execução

```
C:\backupcida\LinguagemCPagina20081\aula18funclinha
*****
Numeros entre 1 e 5
*****
1
2
3
4
5
*****
Pressione qualquer tecla para continuar. . . _
```

# Sintaxe

```
//escrita de numeros inteiros
#include<stdio.h>
#include <stdlib.h>
void apresente_linha(void);
int main( )
{
    int i;
    //apresentacao do cabecalho
    apresente_linha( );
    printf("Numeros entre 1 e 5\n");
    apresente_linha( );
    // Escrita dos numeros
    for (i=1;i<=5;i++)
        printf("%d\n",i);
    apresente_linha( );
    system("pause");
    return 0;
}
```

```
void apresente_linha (void)
```

```
{
    int i;
    for (i=1;i<20;i++)
        printf("*");
    printf("\n");
}
```

Chamadas à função **apresente\_linha** substituem trechos repetidos

Execução

```
C:\ backupcida\LinguagemCPagina20081\aula18funclinhaco
*****
Numeros entre 1 e 5
*****
1
2
3
4
5
*****
Pressione qualquer tecla para continuar. . . _
```

# Sintaxe

//escrita de numeros inteiros

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

```
#include <stdlib.h>
```

```
void apresente_linha(void);
```

```
int main( )
```

```
{
```

```
    int i;
```

```
    //apresentacao do cabecalho
```

```
    apresente_linha( );
```

```
    printf("Numeros entre 1 e 5\n");
```

```
    apresente_linha( );
```

```
    // Escrita dos numeros
```

```
    for (i=1;i<=5;i++)
```

```
        printf("%d\n", i);
```

```
    apresente_linha( );
```

```
    system("pause");
```

```
    return 0;
```

```
}
```

Protótipo da função `apresente_linha`

Chamadas da função `apresente_linha`

Execução

```
C:\backupcida\LinguagemCPagina20081\aula18funclinha.c
*****
Numeros entre 1 e 5
*****
1
2
3
4
5
*****
Pressione qualquer tecla para continuar. . . _
```

```
void apresente_linha (void)
```

```
{
```

```
    int i;
```

```
    for (i=1;i<20;i++)
```

```
        printf("*");
```

```
    printf("\n");
```

```
}
```

Declaração da função

`apresente_linha`:

escreve uma linha de asteriscos.

# Cabeçalho da função `apresente_linha`

`void apresente_linha (void)`



Indica que a função não retorna valor no seu nome.



Indica que a função não tem parâmetros.

A função `apresente_linha` realiza sua tarefa sem receber nenhum valor do mundo externo à função, via parâmetros, e sem retornar nenhum valor no seu nome. Seu tipo é `void` e seus parâmetros são `void`.

# Variáveis locais

Os parâmetros que aparecem no cabeçalho das funções e as variáveis e constantes declaradas internamente a funções são locais à função.

Na função `apresente_linha`, o `i` é uma variável local a essa função.

```
void apresente_linha (void)
{
    int i;
    for (i=1;i<20;i++)
        printf("*");
    printf("\n");
}
```

# Definição de Funções pelo Usuário

- A seguir um programa que calcula o produto de um número indeterminado de pares de valores informados.
- Para calcular os produtos é usada a função definida pelo usuário `calc_produto`;

# produto: função definida pelo usuário (cont.)

```
//calcula produtos de pares de valores informados
#include <stdio.h>
#include <stdlib.h>
int calc_produto(int, int);
main ( )
{
    int seguir;
    int oper1, oper2, produto;
    do
    {
        printf("\nOperando 1: ");
        scanf("%d", &oper1);
        printf("\nOperando 2: ");
        scanf("%d", &oper2);
        printf ("\nProduto = %d\n", calc_produto(oper1, oper2));
        printf("\nPara continuar, digite 1, para parar, digite 0: ");
        scanf("%d", &seguir);
    }
    while (seguir);
    system("pause");
}
```

```
int calc_produto(int valor1, int valor2)
{
    return valor1 * valor2;
}
```

# Quando uma função encerra sua execução?

Uma função encerra sua execução quando:

- o fim do seu código é atingido;

ou

- um comando *return* é encontrado e executado.

# O quê é necessário para usar-se uma função em Linguagem C?

A declaração da função.

A chamada da função.

Dependendo do caso, o protótipo da função.

```
//calcula produtos de pares de valores informados
```

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

```
#include <stdlib.h>
```

```
int calc_produto(int, int);
```

```
main ( )
```

```
{
```

```
int seguir;
```

```
int oper1, oper2, produto;
```

```
do
```

```
{
```

```
printf("\nOperando 1: ");
```

```
scanf("%d", &oper1);
```

```
printf("\nOperando 2: ");
```

```
scanf("%d", &oper2);
```

```
printf ("\nProduto = %d\n" calc_produto(oper1, oper2));
```

```
printf("\nMais um valor, digite 1, para parar, digite 0: ");
```

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

```
}
```

```
while (seguir);
```

```
system("pause");
```

```
}
```

```
int calc_produto(int valor1, int valor2)
```

```
{
```

```
return valor1 * valor2;
```

```
}
```

Protótipo

Chamada da função

Declaração da função

# Exercício

Escreva o código de uma função que calcule a média aritmética de dois valores informados como parâmetros

Escreva um programa que use esta função

```
#include<stdio.h>
#include <stdlib.h>
float media(float, float);
main() {

    float v1,v2,m;
    printf ("Informe os numeros: ");
    scanf("%f %f",&v1,&v2);
    m=media(v1,v2);
    printf("a media dos numeros e': %.4f\n",m);
    system("pause");
}
// declaracao da funcao media
float media(float n1, float n2){
    return((n1+n2)/2);
}
```