

Laboratório de Programação I (MATA57)



Prof.: Claudio Junior N. da Silva (claudiojns@ufba.br)

Funções e Recursão

2023.1

Funções

- Funções (*ou subrotinas, subprogramas*) são blocos de código que agrupam sequências de operações que, atrelados a um “nome”, podem ser invocados em todo o programa, de forma que tal sequência possa ser executada;
- A ideia de funções está em encapsular uma ideia ou operação;
- **Benefícios:**
 - Clareza do Código
 - Reutilização
 - Independência

Declaração de uma função

```
[tipo de retorno da função] [nome da função] (1º parâmetro, 2º parâmetro, nº  
    parâmetro) {  
    //corpo da função  
}
```

Declaração de uma função

- Em C/C++, todas as funções **DEVEM** ser definidas **ANTES** da função **main()**, ou então utilizar um protótipo da função;
- O nome de uma função DEVE ser **ÚNICO** (não pode ser igual ao nome de outra função ou de uma variável);
- A mesma regra de nomenclatura de variáveis é aplicável à nomenclatura de funções

Valor de retorno

- Uma função pode executar algum tipo de processamento (ou cálculo) e precisa retornar o resultado dessa operação.

[tipo de retorno da função] >>>> [int, float, double, char, void]

Valor de retorno - void

- Se o retorno for `void` significa que a função se comporta como uma subrotina, i.e. a função não precisa retornar um valor - ex: `printf()`

Muitos dizem que a função `main()` tem tipo de retorno `void`, o que não está correto.

Segundo o padrão da linguagem C, a função `main()` deve ter retorno do tipo `int`.

Alguns compiladores, a exemplo do `gcc`, dão mensagens de erro caso a função `main()` não seja definida corretamente.

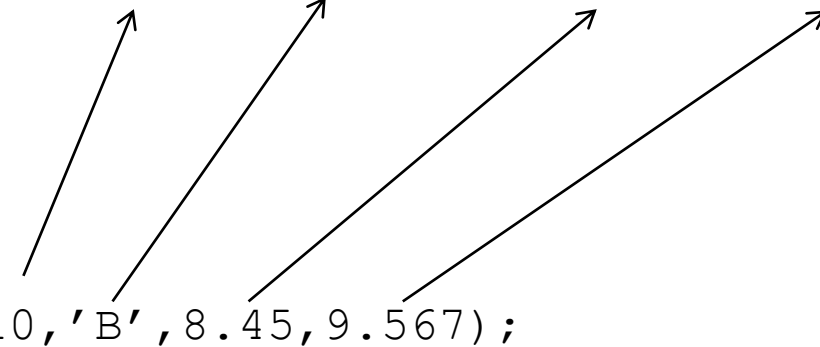
Parâmetros

- A comunicação com uma função se faz através dos **argumentos** que lhe são enviados e dos **parâmetros** presentes na função que os recebe
- Os parâmetros funcionam como variáveis locais

Ex: `void funcao(int a, char b, float c, double d)`

- Quando se faz a chamada de uma função, o número e o tipo dos argumentos enviados devem ser coincidentes com os parâmetros presentes no cabeçalho de função.

```
void funcao(int a, char b, float c, double d) {  
    ...  
}  
  
main() {  
    ...  
    funcao(10, 'B', 8.45, 9.567);  
    ...  
}
```

A diagram with four arrows pointing from the arguments in the function call to the parameters in the function signature. The first arrow points from '10' to 'int a'. The second arrow points from ''B'' to 'char b'. The third arrow points from '8.45' to 'float c'. The fourth arrow points from '9.567' to 'double d'.

Protótipo de uma Função

- Caso a definição da função apareça após a função main(), o seguinte erro virá a tona:

function <função>: redefinition;

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

```
void teste(); //declaração da função teste()
```

```
int main() {
```

```
    ...
```

```
    teste(); /*invocação à função teste, dentro do      bloco principal*/
```

```
    ...
```

```
}
```

```
void teste() {
```

```
    //corpo da função teste
```

```
}
```


Exemplo de uma função

Tipo de retorno

Parâmetros

```
int soma(int numA, int numB) {  
    int resultadoSoma;  
    resultadoSoma = numA + numB;  
    return resultadoSoma;  
}
```

Valor de retorno

The diagram illustrates the parts of the function signature `int soma(int numA, int numB)`. The label 'Tipo de retorno' (Return type) points to the `int` at the start. The label 'Parâmetros' (Parameters) points to the two `int` types followed by `numA` and `numB`. The label 'Valor de retorno' (Return value) points to the `return` statement inside the function body.

A função `main()`

- Um programa em C deve possuir SEMPRE a função `main()` escrita no seu código, independentemente das outras funções existentes no programa.

```
#include<stdio.h>

int main (int argc, char **argv) {

    ...

}
```

Argumentos da função `main()`

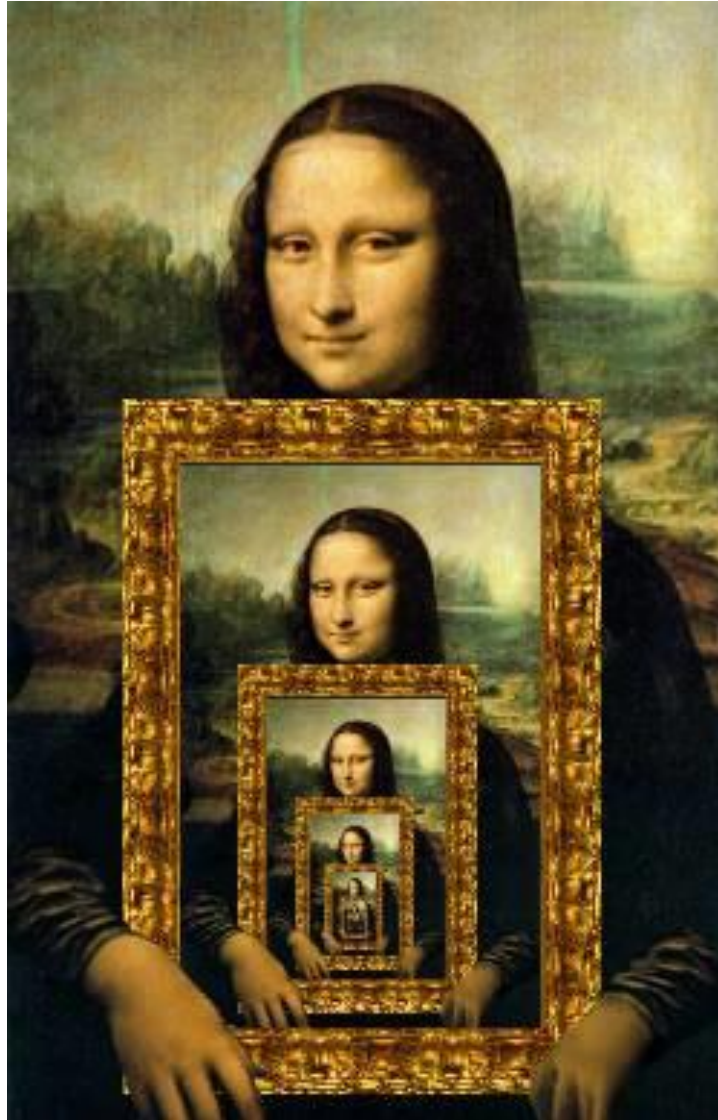
- **argc**: é um inteiro que indica o número de argumentos digitados na linha de comando;
- **argv**: é um ponteiro para um *string* (sequência de caracteres) que contém todos os argumentos da linha de comando.

```
int main (int argc, char **argv) {  
    if(argc != 2){  
        printf("voce esqueceu de digitar seu nome\n");  
    }  
    printf("alo %s", argv[1]);  
    return 0;  
}
```

Exercício 15 – Cálculo da Média

- Implemente um programa em C++ que:
 - Leia quatros notas de um aluno;
 - Faça o calculo da média executando uma função chamada `calc_media`;
 - Imprima o resultado do cálculo.

Recursividade



Conceito de recursividade

- Um objeto é dito **recursivo** se ele consistir parcialmente ou for definido em termos de **si próprio**;
- Recursão é o processo **repetitivo** no qual uma função chama a **si mesma**:
 - Chamada direta ou indireta
- Uma função recursiva deve seguir duas regras básicas:
 - Ter uma condição de parada;
 - Tornar o problema mais simples.
- Quando uma função recursiva está sendo executada, são **alocados novos parâmetros** e **variáveis locais** na **memória**, para **cada** chamada da função em questão

Recursividade na programação

- Vantagens:
 - Simplificam a solução de alguns problemas;
 - Redução do código fonte da rotina.
- Desvantagens
 - São mais lentas que funções iterativas, devido ao tempo gasto no gerenciamento;
 - Erros de implementação podem levar à estouro de pilha (Pilha de Execução)
 - Dificuldade de depuração dos programas, principalmente se a recursão for muito profunda

Recursividade na programação

- Para escrever uma rotina recursiva, devemos definir dois elementos:
 - uma condição de parada (escape/caso base);
 - a expressão recursiva (o próprio módulo chama a si mesmo).
- A linguagem C permite a recursão
 - Gerencia a passagem de parâmetros quando estes subprogramas são chamados, usando uma estrutura de dados especial chamada **Pilha de Execução (*stack*)**;

Funções Recursivas - Fatorial

Fatorial N:

| $N = 0, 1$

| $N > 0, N * \text{Fatorial}(N-1).$

Fatorial(5):

$\Rightarrow 5 * \text{Fatorial}(4) \Rightarrow 5 * 24 = 120$

$\Rightarrow \text{Fatorial}(4) : 4 * \text{Fatorial}(3) \Rightarrow 4 * 6 = 24$

$\Rightarrow \text{Fatorial}(3) : 3 * \text{Fatorial}(2) \Rightarrow 3 * 2 = 6$

$\Rightarrow \text{Fatorial}(2) : 2 * \text{Fatorial}(1) \Rightarrow 2 * 1 = 2$

$\Rightarrow \text{Fatorial}(1) : 1 * \text{Fatorial}(0) \Rightarrow 1 * 1 = 1$

$\Rightarrow \text{Fatorial}(0) : 1$

$5 * (4 * (3 * (2 * (1 * 1))))$

$5 * (4 * (3 * (2 * 1)))$

$5 * (4 * (3 * 2))$

$5 * (4 * 6)$

$5 * 24$

120

Uma solução recursiva para um problema envolve um caminho de dois sentidos:

1) o problema é decomposto no sentido top/down e 2) resolvido no sentido bottom/up

Exercício 16 – Cálculo do Fatorial

Implemente um programa em C++ que:

- Leia um número inteiro;
- Execute uma função recursiva para calcular o fatorial do número informado;
- Apresente o resultado;

- Função fatorial

- $n! =$

- 1, se $n=0$ ou $n=1$
 - $n * (n-1)!$, se $n>1$

Exercício 16 – Cálculo do Fatorial

```
#include <iostream>
using namespace std;

int calcularFatorial_Rec(int num) {
    int fatorial;
    if(num == 0)
        fatorial = 1;
    else
        fatorial = num * calcularFatorial_Rec(num-1);
    return fatorial;
}

int main() {
    int num;
    cout << "Informe um número inteiro e positivo: ";
    cin >> num;
    cout << "O fatorial de " << num << " eh: " <<
    calcularFatorial_Rec(num) << endl;
    return 0;
}
```

Exercício 17 – Elevar um número a uma Potência

Implemente um programa em C++ que:

- Leia um número inteiro maior do que ZERO;
- Leia um expoente com valor igual ou maior a 0;
- Execute uma função recursiva para calcular o número elevado ao expoente informado.
- Potencia(N,E) :
 - 1, se $E = 0$;
 - $N * \text{Potencia}(N, E - 1)$.

Exercício 17 – Elevar um número a uma Potência

```
#include <iostream>
using namespace std;

int calc_potencia(int numero, int e){
    int resultado;
    if(e == 0)
        resultado = 1;
    else
        resultado = numero * calc_potencia(numero, e-1);
    return resultado;
}

int main(){
    int numero, e, resultado;
    cout << "Informe um numero inteiro: " << endl;
    cin >> numero;
    cout << "Informe a potencia: " << endl;
    cin >> e;
    cout << "O resultado eh: " << calc_potencia(numero, e) << endl;
}
```

Exercícios – Sala de aula

18. Crie um programa em C++ que leia os preços de 4 produtos e execute uma função para calcular a média dos preços dos produtos;

18. Crie um programa em C++ para calcular as médias das alturas e dos pesos de 4 pessoas:

- Entrada: seu programa deve ler as medidas das alturas das 4 pessoas. Em seguida deve ler as medidas dos pesos das quatro pessoas;
- Saída: seu programa deve calcular as médias das alturas e dos pesos das pessoas e em seguida apresentar o resultado:
 - A média de altura é: ??.
 - A média de peso é: ??.