

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa



Tecnologia em Sistemas para Internet

Linguagem “C”

Funções

valeriacavalcanti.com.br

Profa. Valéria Cavalcanti

valeria.cavalcanti@ifpb.edu.br



Macro

- São operações definidas por símbolos;
- São compiladas da mesma forma das constantes simbólicas;
- Podem ser definidas com ou sem parâmetros;
- Argumentos de macros **não são variáveis**, portanto, não tem tipo nem memória alocada. Uma única macro pode servir para mais de um tipo de dado;
- Macros são mais rápidas do que as funções.

Macro

Exemplos:

- `#define SOMA(a, b) (a + b)`
- `#define MULT(a, b) (a * b)`
- `#define QUAD(x) (x * x)`
- `#define out printf`
- `#define in scanf`
- `#define MAIOR(a, b) ((a > b) ? (a) : (b))`
- `#define MENOR(a, b) ((a < b) ? (a) : (b))`

Macro: Exemplos

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

```
#define SOMA(a, b) (a+b)
```

```
#define MAIOR(a, b) ((a > b) ? (a) : (b))
```

```
#define out printf
```

```
int main() {
```

```
    printf("Soma = %d\n", SOMA(10, 20));
```

```
    printf("Maior = %d\n", MAIOR(10, 20));
```

```
    printf("Maior = %d\n", MAIOR(20, 10));
```

```
    printf("Maior = %d\n", MAIOR(20, 20));
```

```
    out("Funciona!");
```

```
    return 0;
```

```
}
```

Macro

Alguns erros ...

- `#define if se`
- `#define TAM = 10`
- `#define TAM = 10;`

Complexidade → Solução



- Se ao desenvolver software o programador for pensar em todos os possíveis e prováveis problemas que ele terá que resolver ...
sai nada !!

Analogia: Organizar uma festa !!



A única coisa pronta é o dinheiro e a lista de convidados!

Analogia: Organizar uma festa !!

- Comprar bebidas;
- Comprar salgadinhos;
- Comprar o jantar;
- Comprar os fogos;
- Organizar o som (músicas);
- Organizar os garçons;
- Organizar a infra-estrutura (mesa e cadeira);
- ...



Estratégia

É preciso montar um plano para
não desistir da festa !



Xeque-mate!

Montando estratégia

- Ao pensar em **cada tarefa**, devemos analisar ...



- O que é necessário **saber** (dados) ?
- O que vou **fazer** com os dados informados ?
- Devo alguma **resposta** ?

Analizando algumas tarefas

T001: Comprar bebidas.

- O que é necessário saber (dados) ?

Qtde de convidados e orçamento.

A diagram consisting of an orange line that starts from the right side of the text 'Qtde de convidados e orçamento.', goes horizontally to the right, then vertically down, and finally horizontally left into the right side of an orange rounded rectangle.

- O que vou fazer com os dados informados ?

Comprar bebidas de acordo com o nº de convidados e orçamento disponível.

- Devo alguma resposta ?

Não !

Esses dados são
realmente
necessários?!

Analizando algumas tarefas

Sem dúvida!! Vai ajudar muito a decidir quais produtos serão comprados !!



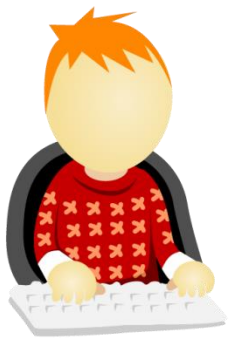
Pouca verba para o número de convidados.

Muita verba para o número de convidados.

Função

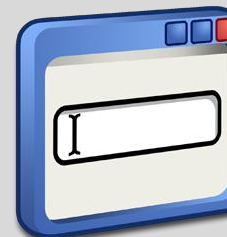
- Peça chave da programação estruturada;
- Um fragmento com **começo, meio e fim**, que desempenha um papel **bem específico** dentro de um programa maior;
- Essas funções podem ser implementadas **separadamente** e **por diversos programadores** de uma equipe.

Arquitetura



main ()

funções ()



Função

- **Sintaxe:**

```
tipo_retorno nome (parâmetros){  
    <comandos>  
}
```

- **Curiosidade:**

Uma função pode conter variável exclusiva dela ?

Resposta: **Sim !!** São as variáveis locais.

Parâmetros

- Valores que uma função necessita para executar a sua tarefa;

Tipos:

- Variável comum ou apontador.

Classificação:

- Formal e real.

Exemplos

Exemplo 1

- Função para somar dois números:

```
int somar (int n1, int n2) {  
    return n1 + n2;  
}
```

Exemplo 1

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

```
int somar (int n1, int n2);
```

```
int main(){  
    int num1 = 10, num2 = 20;
```

```
    printf("Soma = %d\n", somar(num1, num2));
```

```
    return 0;
```

```
}
```

```
int somar (int n1, int n2){  
    return n1 + n2;  
}
```

Assinatura da função

Chamada (uso)

Parâmetro real

Passagem de parâmetros por valor

Parâmetro formal

Declaração (implementação)

Exemplo 2

- Função para subtrair dois números:

```
int subtrair (int n1, int n2) {  
    return n1 - n2;  
}
```

Exemplo 2

```
#include <stdio.h>

int subtrair (int n1, int n2);

int main() {
    int num1 = 10, num2 = 20;

    printf("Subtracao = %d\n", subtrair(num1, num2));

    return 0;
}

int subtrair (int n1, int n2) {
    return n1 - n2;
}
```

Exemplo 3

- Função para, de acordo com a operação definida, ou somar, ou subtrair:

```
int somar (int n1, int n2) {  
    return n1 + n2;  
}
```

```
int subtrair (int n1, int n2) {  
    return n1 - n2;  
}
```

```
int operacao (char tipo, int n1, int n2) {  
    if (tipo == '+') return somar(n1, n2);  
    else if (tipo == '-') return subtrair(n1, n2);  
    else return 0;  
}
```

Exemplo 3

```
#include <stdio.h>

int somar (int n1, int n2);
int subtrair (int n1, int n2);
int operacao (char tipo, int n1, int n2);

int main(){
    int num1 = 10, num2 = 20;

    printf("Somar = %d\n", operacao('+', num1, num2));
    printf("Subtrair = %d\n", operacao('-', num1, num2));

    return 0;
}

int somar (int n1, int n2){
    return n1 + n2;
}

int subtrair (int n1, int n2){
    return n1 - n2;
}

int operacao (char tipo, int n1, int n2){
    if (tipo == '+') return somar(n1, n2);
    else if (tipo == '-') return subtrair(n1, n2);
    else return 0;
}
```

Exemplo 4

- Função calcular a quantidade de divisores de um determinado número:

```
int divisores (int n) {  
    int i, qtde = 0;  
  
    for (i = 2; i <= n/2; ++i)  
        if (n % i == 0) ++qtde;  
  
    if (n == 1) return 1;  
    else return qtde + 2;  
}
```


Exemplo 4

```
#include <stdio.h>

int divisores (int n);

int main(){

    printf("Qtde divisores (10) = %d\n", divisores(10));
    printf("Qtde divisores (6) = %d\n", divisores(6));
    printf("Qtde divisores (100) = %d\n", divisores(100));
    printf("Qtde divisores (1) = %d\n", divisores(1));
    printf("Qtde divisores (2) = %d\n", divisores(2));

    return 0;
}

int divisores (int n){
    int i, qtde = 0;

    for (i = 2; i <= n/2; ++i)
        if (n % i == 0) ++qtde;

    if (n == 1) return 1;
    else return qtde + 2;
}
```

Atenção!

- Todos os exemplos foram implementados realizando a passagem de parâmetros “por valor”, ou seja, apenas são passados os valores (cópia) das variáveis (parâmetros reais);
- Na passagem de parâmetros “por referência” é passado uma referência (endereço) para o parâmetro formal. Dessa forma, a função consegue realizar alterações no parâmetro.

Exemplo 5

- Função para trocar os valores entre duas variáveis:

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

Exemplo 5

```
#include <stdio.h>

void troca (int *n1, int *n2);


int main() {
    int num1 = 10, num2 = 20;

    printf("%d %d\n", num1, num2);
    troca(&num1, &num2);
    printf("%d %d\n", num1, num2);

    return 0;
}

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

Na passagem de parâmetros por referência, são passados os endereços das variáveis !



Exemplo 6

- Função para imprimir um vetor de inteiros:

```
void print_vetor (int vetor[], int tam) {  
    int i;  
  
    for (i = 0; i < tam; ++i)  
        printf("%d ", vetor[i]);  
  
    printf("\n");  
}
```

Exemplo 6

```
#include <stdio.h>

void print_vetor (int vetor[], int tam);

int main() {
    int numeros[] = {10, 20, 30, 40, 50, 60};

    print_vetor(numeros, 6);
    print_vetor(numeros, 4);

    return 0;
}

void print_vetor (int vetor[], int tam) {
    int i;

    for (i = 0; i < tam; ++i)
        printf("%d ", vetor[i]);

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