AULA 05

{introcomp}

MODULARIZAÇÃO

Modularização

Porque modularizar um código?

-Decompor uma tarefa complexa em tarefas menores e de fácil solução.

-Fazer uso da técnica “dividir para conquistar”

-Evitar repetição de código

Modularização

Vantagens:

-Boa legibilidade de código;

-Facilidade em manutenção;

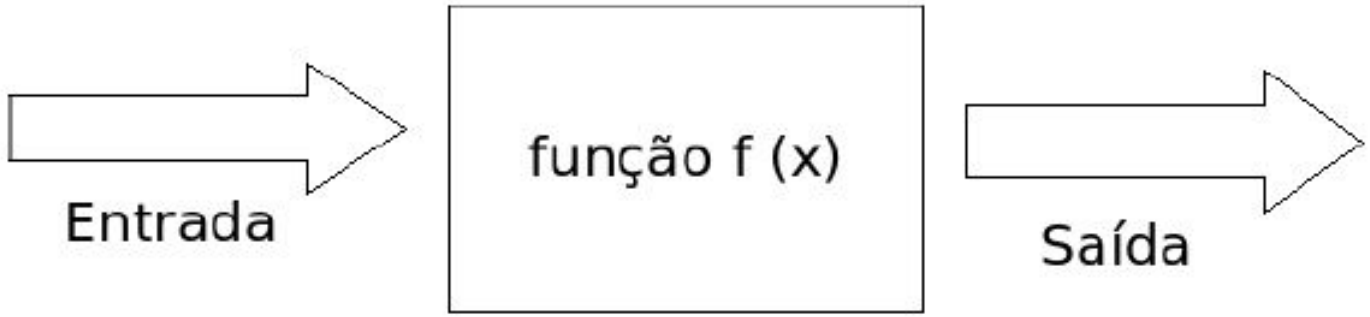
-Confiabilidade do programa;

-Reutilização de código.

Definição

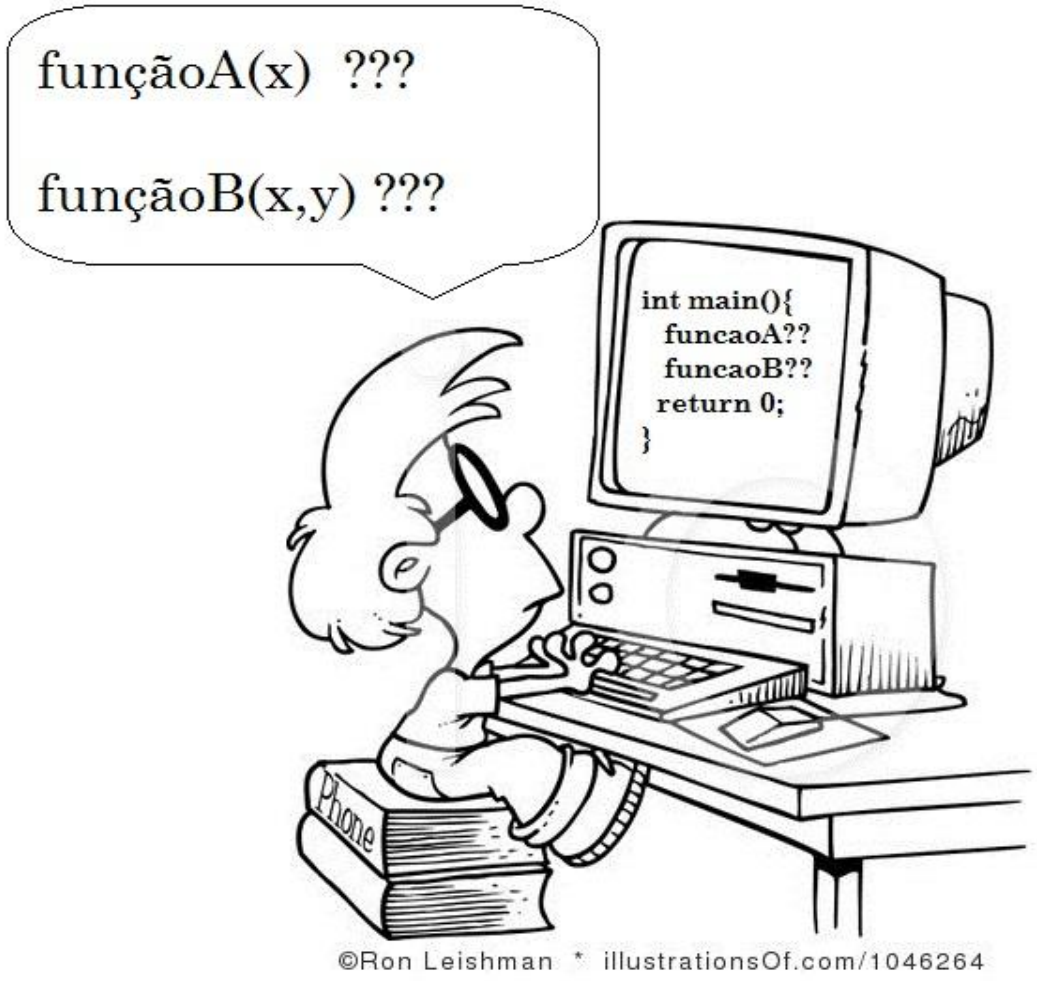
Entrada: Parâmetros;

Saída: Retorno;



Função

Como construir uma função em C?



Estrutura de uma função

Sintaxe:

-Tipo do retorno **(apenas um tipo de retorno)**;

-Nome;

-Tipos dos parâmetros de entrada;

Tipos de retorno

Os tipos de retorno possíveis são:

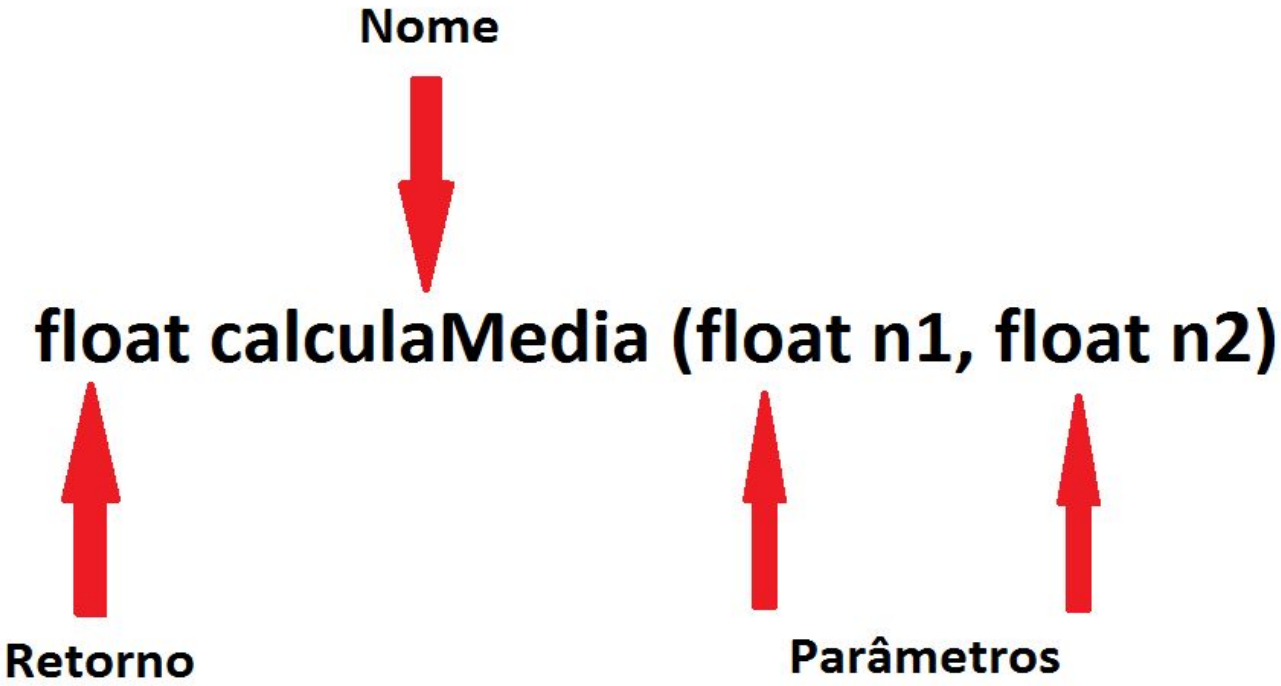
-**int**: retorna um valor inteiro

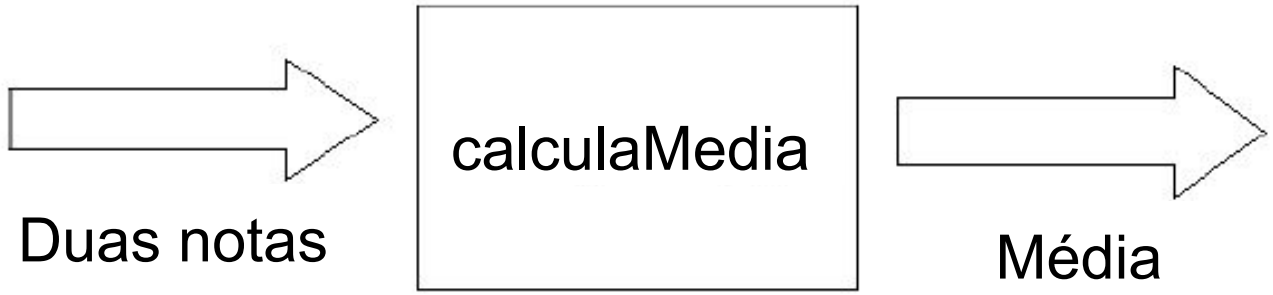
-**float, double**: retorna valores reais

-**char**: retorna um caractere

-**void**: função sem retorno

Estrutura de uma função





Implementação

```
1float calculaMedia(float n1,float n2){
2
3     float media;
4     media = (n1+n2)/2;
5     return media;
6}
```

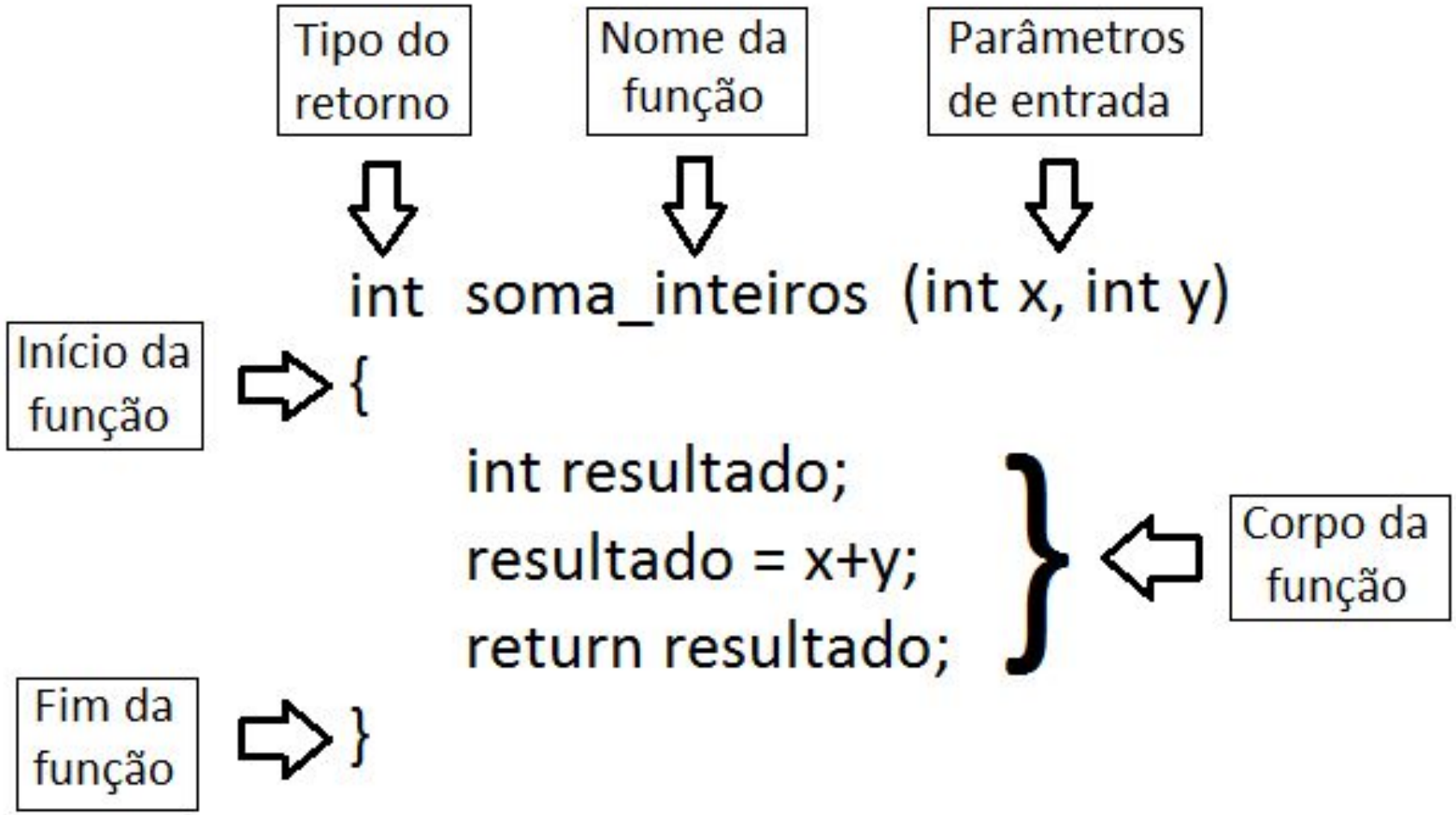
Como chamar uma função

```c
#include <stdio.h>

float calculaMedia(float n1,float n2){

    float media;
    media = (n1+n2)/2;
    return media;
}

int main(){

    float n1,n2,media;
    scanf("%f %f",&n1,&n2);
    media = calculaMedia(n1,n2);

    printf("A media eh: %f",media);
    return 0;
}
```

Estrutura de uma função - exemplos

Faça uma função que some 2 variáveis inteiras.

Estrutura de uma função – exemplos



Estrutura de uma função - exemplos

Faça uma função que **imprima** um intervalo fechado [x,y].

Estrutura de uma função

```
1 void printa_intervalo(int x, int y){
2     int i = 0;
3     for(i = x; i <= y; i++){
4         printf("%d ", i);
5     }
6     printf("\n");
7 }
8
```

Estrutura de uma função - exemplos

Faça uma função que retorne um inteiro lido pelo teclado.

Estrutura de uma função

```
int retorna_lido(){
        int a;
        scanf("%d", &a);
        return a;
}
```

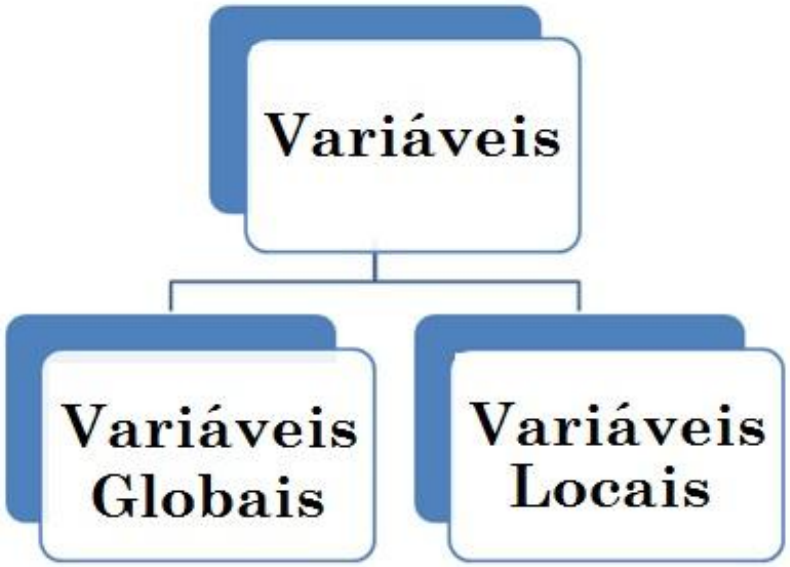
Tipos de retorno

Observe a função main, ela retorna “0” (inteiro) ao fim da execução do programa e não tem parâmetro de entrada (void)

```c
#include <stdio.h>

int main(void){

return 0;
}
```

Variáveis

Variáveis locais e globais:




```c
#include <stdio.h>

int a = 1;  //variável global

int main(){
    int b = 2; //variável local

    printf("Valor da variavel 'a': %d", a);
    printf("\nValor da Variavel 'b': %d", b);

return 0;
}
```

Variáveis – Variável global x Variável Local

Variável Global

- Pode ser vista por **todas** as funções.
- **Não** acaba quando uma função termina.

Variável Local

- Só pode ser vista por **uma** função.
- Acaba quando a função termina.

Obs: Todos os valores alterados no escopo de uma função, somente surtirão efeito neste escopo. Ou seja, caso você queira modificar o conteúdo de uma variável através de uma função, este valor deverá ser retornado e atribuído!

Variáveis

```
int main()
{
      int valor = 2; // variavel local
      valor = valor + 2;
      printf("%d",valor);
}
```

Escopo da variável local

```
void add()
{
      valor = valor + 4;
      printf("%d",valor);
}
```

A variável local *valor* não pode ser utilizada nesta função.

Funções - Laboratório

Faça um programa que leia 2 números inteiros do teclado e calcule o produto e a diferença entre eles.

Obs: Faça duas funções, uma para calcular o produto e outra para calcular a diferença.

Funções - Laboratório

Faça um programa que dado os valores do peso e da altura de um indivíduo, calcule o seu IMC e retorne sua classificação, de acordo com a tabela abaixo:

IMC	Classificação
abaixo de 18,5	abaixo do peso
entre 18,6 e 24,9	Peso ideal (parabéns)
entre 25,0 e 29,9	Levemente acima do peso
entre 30,0 e 34,9	Obesidade grau I
entre 35,0 e 39,9	Obesidade grau II (severa)
acima de 40	Obesidade III (mórbida)

Para o cálculo do IMC, utilize a fórmula **IMC = Peso/Altura²**.

Obs: Crie uma função que receba como parâmetro o peso e a altura e imprima, de acordo com o cálculo do IMC, a classificação.

A impressão deverá ser feita **dentro** da função!!

AULA 05

{introcomp}

MODULARIZAÇÃO