AULA 05

{introcomp}

MODULARIZAÇÃO

O que é a modularização

Código Modularizado

```c
#include <stdio.h>

int ParOuImpar (int numero)
{
    if(numero%2==0)
        return 1;
    else
        return 0;
}

void Resposta(int numero1,int numero2)
{
    if(numero1==1 && numero2==1)
        printf("Ambos são pares\n");
    else
        printf("Ambos não são pares\n");
}
int main(){

    int a,b,c,d;

    printf("Insira 4 números\n");
    scanf("%d %d %d %d",&a,&b,&c,&d);

    Resposta(ParOuImpar(a),ParOuImpar(b));
    Resposta(ParOuImpar(c),ParOuImpar(d));

    return 0;

}
```

Código Não Modularizado

```c
#include <stdio.h>

int main(){

    int a,b,c,d;

    printf("Insira 4 números\n");
    scanf("%d %d %d %d",&a,&b,&c,&d);

    if(a%2==0)
        a=1;
    else
        a=0;

    if(b%2==0)
        b=1;
    else
        b=0;

    if(c%2==0)
        c=1;
    else
        c=0;

    if(d%2==0)
        d=1;
    else
        d=0;

    if(a==1 && b==1)
        printf("Ambos são pares\n");
    else
        printf("Ambos não são pares\n");

    if(c==1 && d==1)
        printf("Ambos são pares\n");
    else
        printf("Ambos não são pares\n");

    return 0;

}
```

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.

Utilização de funções

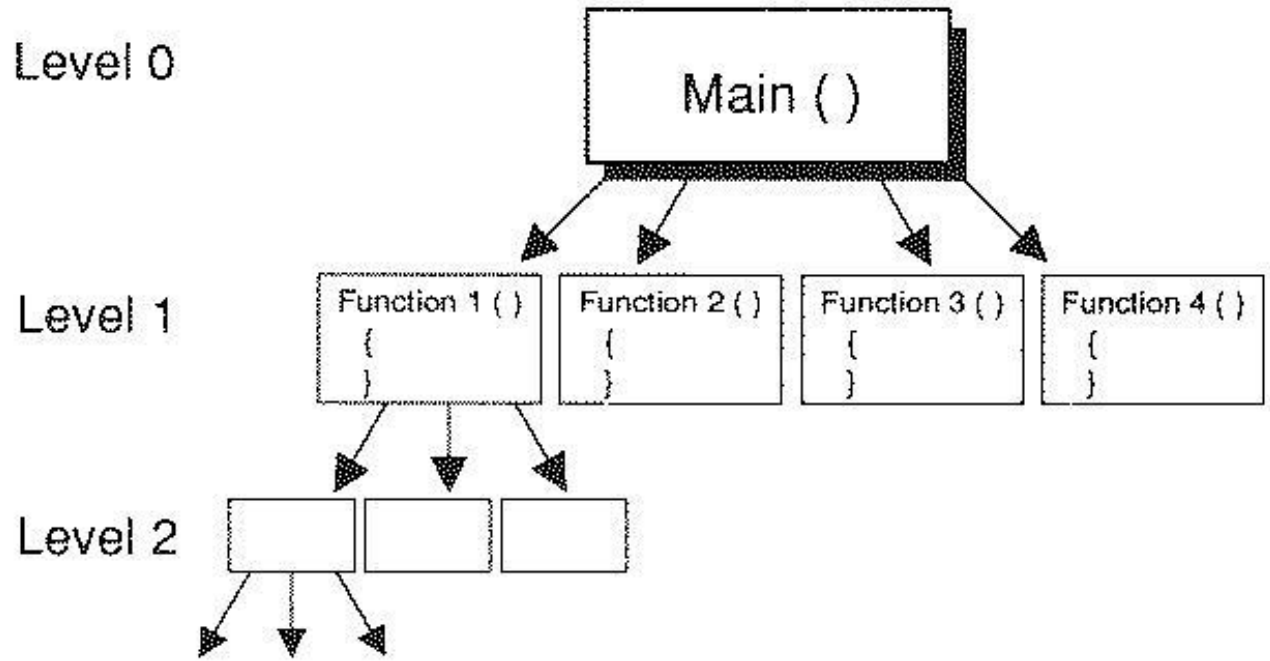


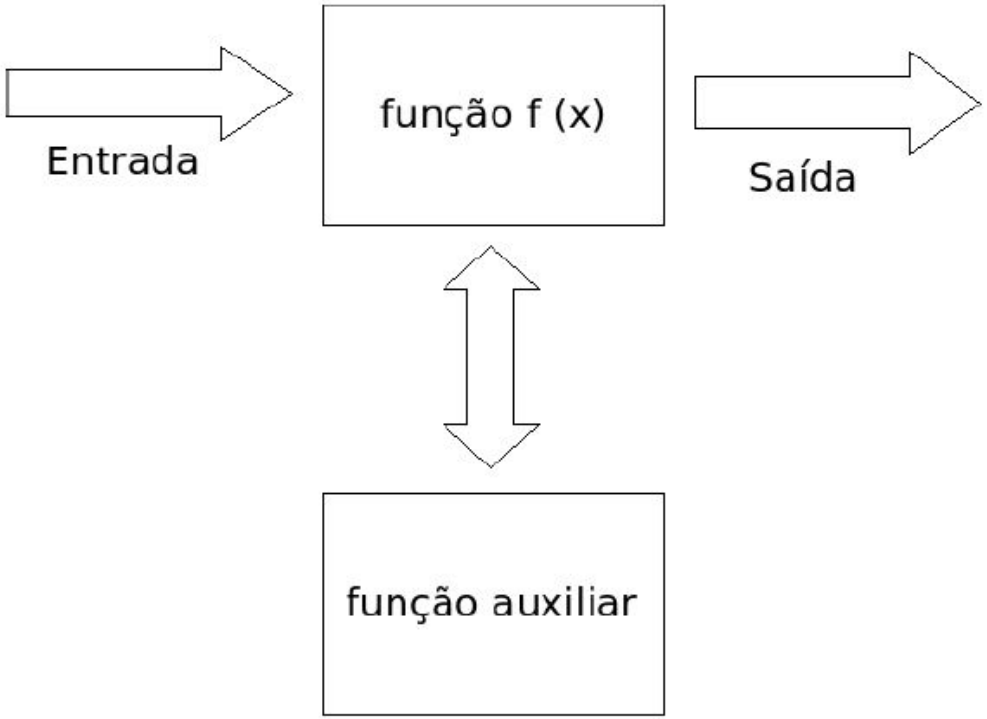
Figure 7.1. C Programs are built from functions.

Utilização de funções

Entrada: Parâmetros;

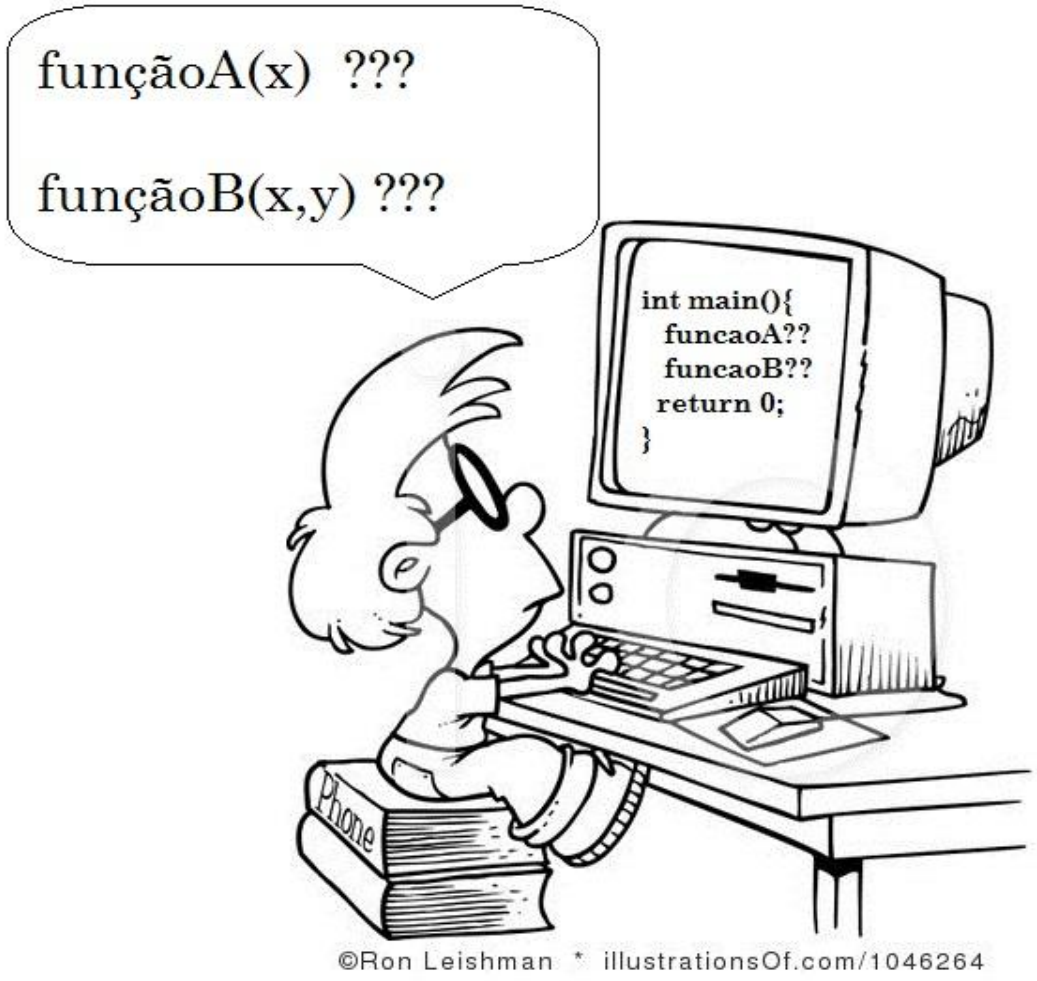
Saída: Retorno;

-Dentro de uma função, pode-se usar funções auxiliares.



Função

Como construir uma função em C?



Estrutura de uma função

Cabeçalho:

-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 carácter

-void: função sem retorno

-retornos compostos, ex: long int, unsigned char, etc

Tipos de retorno

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

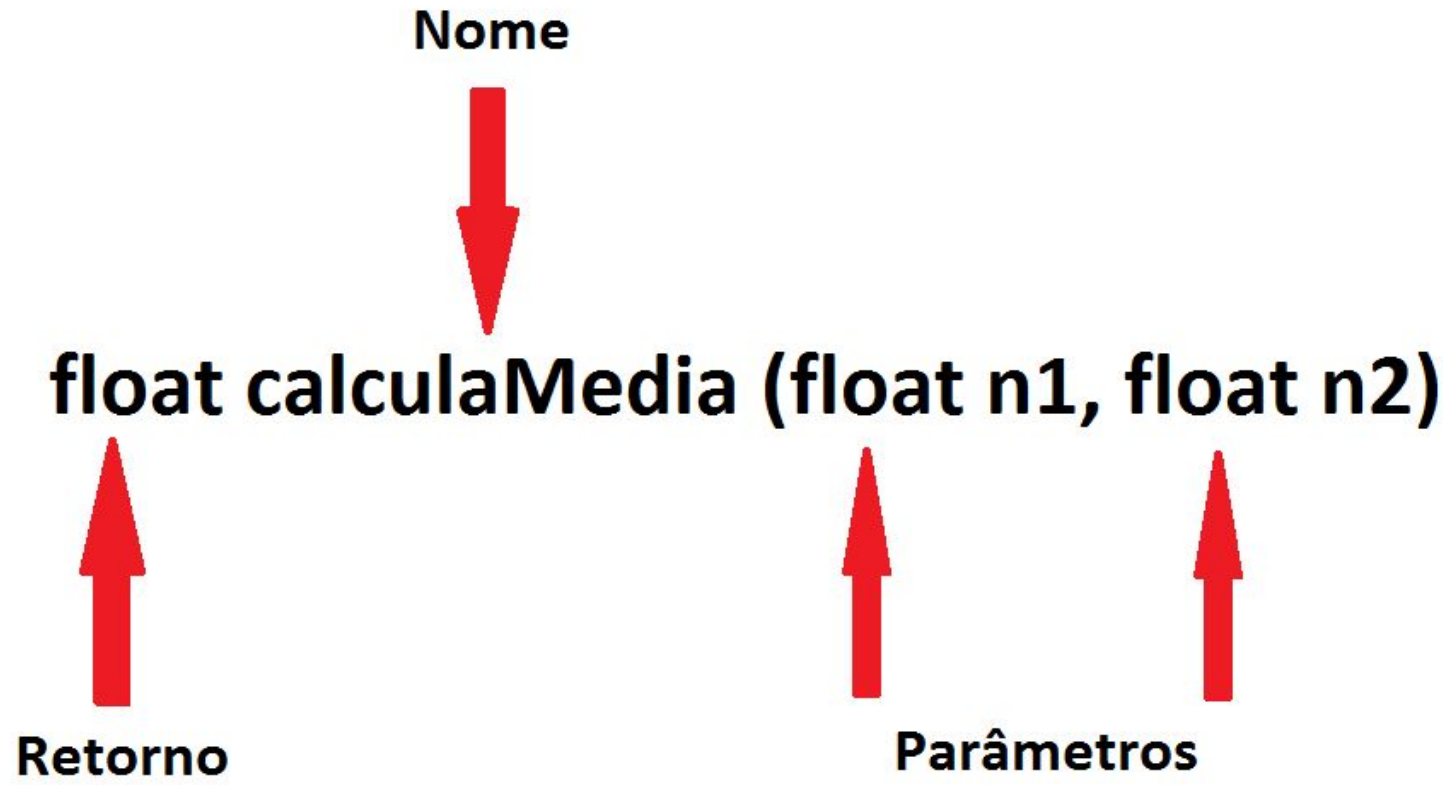
```
#include <stdio.h>

int main (void){

   return 0;
}
```

Estrutura de uma função

Exemplo de cabeçalho:



Estrutura de uma função

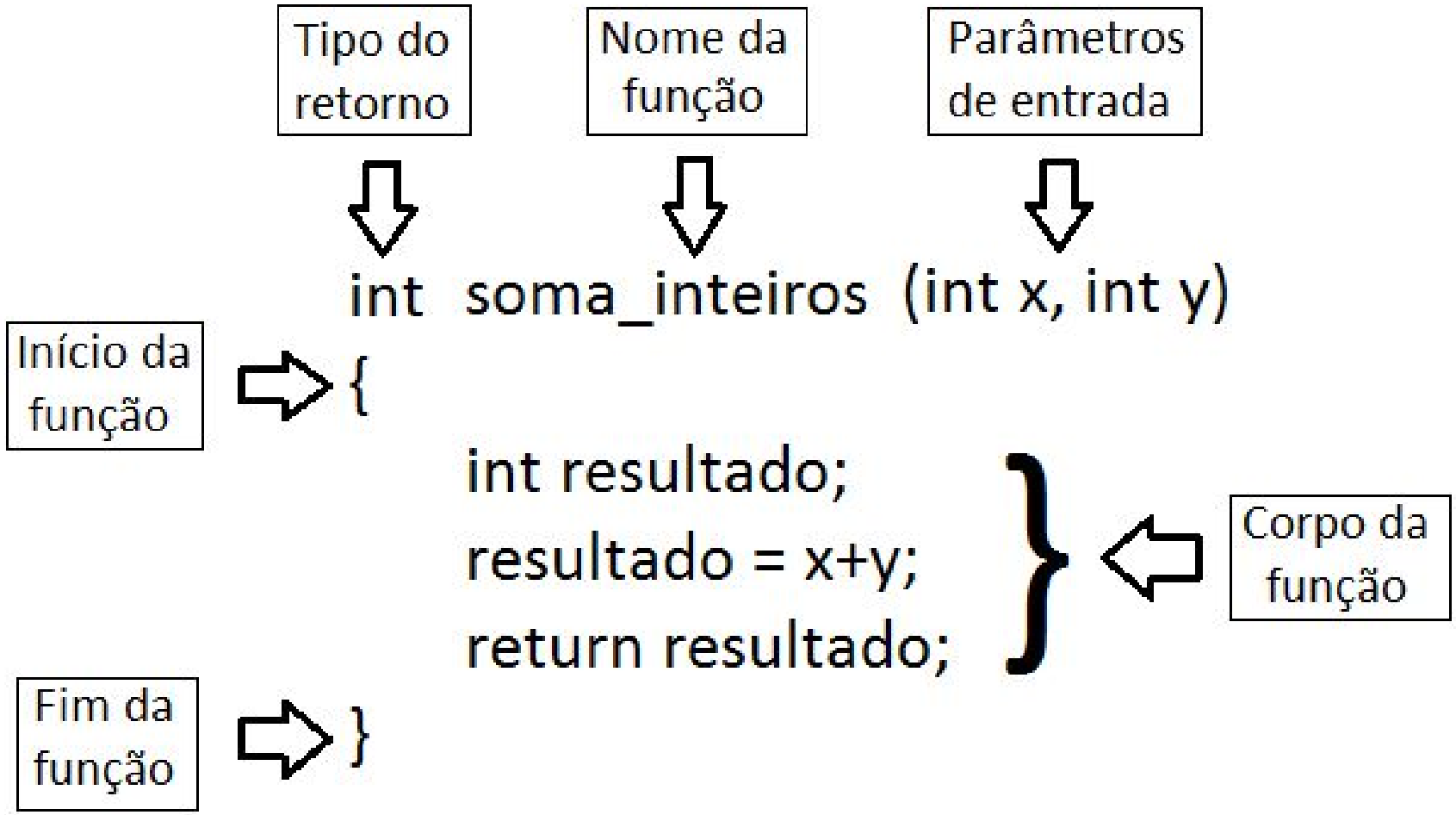
Como chamar uma função:

```
int main ()
{
   float n1=0,n2=0,media=0;
   scanf("%f %f",&n1,&n2);
   media = calculaMedia(n1,n2);
   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

```
void printa_intervalo(int x, int y)
{
   int i=0;
   for(i=x;i<=y;i++)
   {
      printf("%d ", i);
   }
   printf("\n");

}
```

Estrutura de uma função - exemplos

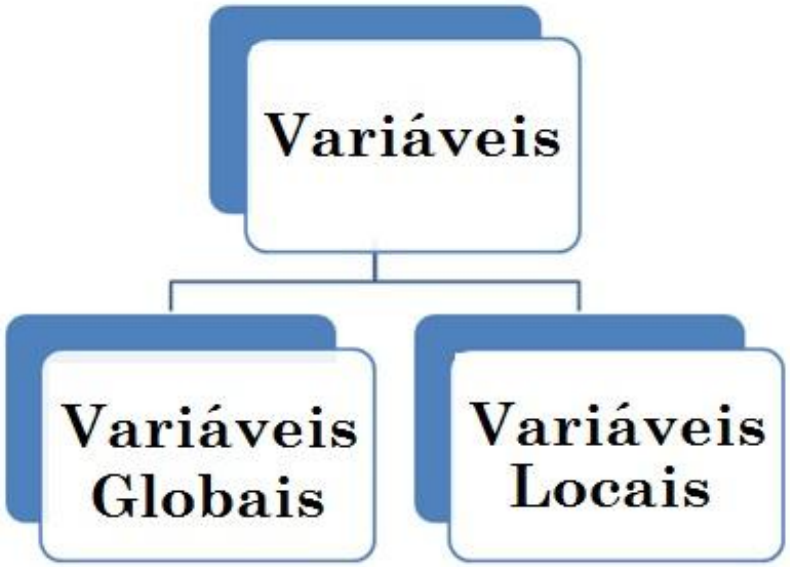
Faça uma função que retorne um inteiro lido pelo teclado.

Estrutura de uma função

```
int retorna_lido()
{
   int a=0;
   scanf(“%d”,&a);
   return a;
}
```

Variáveis

Variáveis locais e globais:




```
#include <stdio.h>

int a; // Variavel Global

int main (){

    int  b = 2; // Variavel Local
    printf (“Variavel a: %d”, a);
    printf (“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.

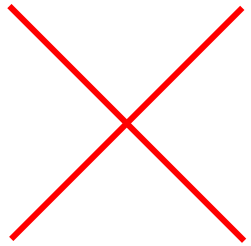
Obs: Todos os valores alterados no escopo de uma função, somente surtirão efeito neste escopo. Ou seja, caso você queira modificar o conteúdo de uma variável através de uma função, este valor deverá ser retornado e atribuído!

Variáveis

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

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

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



Função Recursiva

Objetivo:

- Uma função recursiva é uma função que chama ela própria.
- Esse tipo de função requer um **critério de parada**.
- Possui grande semelhança com as estruturas de repetição.

Função Recursiva

Um exemplo comum de função recursiva é a sequência de Fibonacci. A própria definição da sequência de Fibonacci pode ser tomada como base para implementar um algoritmo recursivo que gera os termos da sequência, como mostrado a seguir:

$$F(n) = \begin{cases} 0, & \text{se } n = 0; \\ 1, & \text{se } n = 1; \\ F(n-1) + F(n-2) & \text{outros casos.} \end{cases}$$

Função Recursiva - exemplos

```
#include <stdio.h>
int fibonacci (int num){
    if (num==0){
        return 0;
    }
    if (num==1){
        return 1;
    }
    return (fibonacci(num-1) + fibonacci (num-2));
}

int main (){

    int n; i;
    scanf (“%d”, &n);

    for ( i=1; i <= n; i++){

        printf (“%d”, fibonacci(i) );

    return 0;
}
```

Funções - Laboratório

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

Obs: Faça uma função para calcular essa média.

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!!

Funções - Laboratório

Faça um programa que leia um número **N** do teclado e printe os termos (N, N - 2, N - 4, ...) enquanto esses números forem positivos.

Obs: Faça uma função recursiva para calcular esses termos.

Funções - Laboratório

Faça uma programa que leia um inteiro **N** e mostre na tela o fatorial de **N**.

Obs: Utilize função recursiva.
Obs 2: Lembrem-se que o fatorial de um número nada mais é que:

Fatorial(N) = N*(N-1)*(N-2)*(N-3)*...*(N-N)!, sendo que o fatorial de (N-N)! é sempre 1! (Que é o mesmo que fatorial de zero)

AULA 05

{introcomp}

MODULARIZAÇÃO