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;

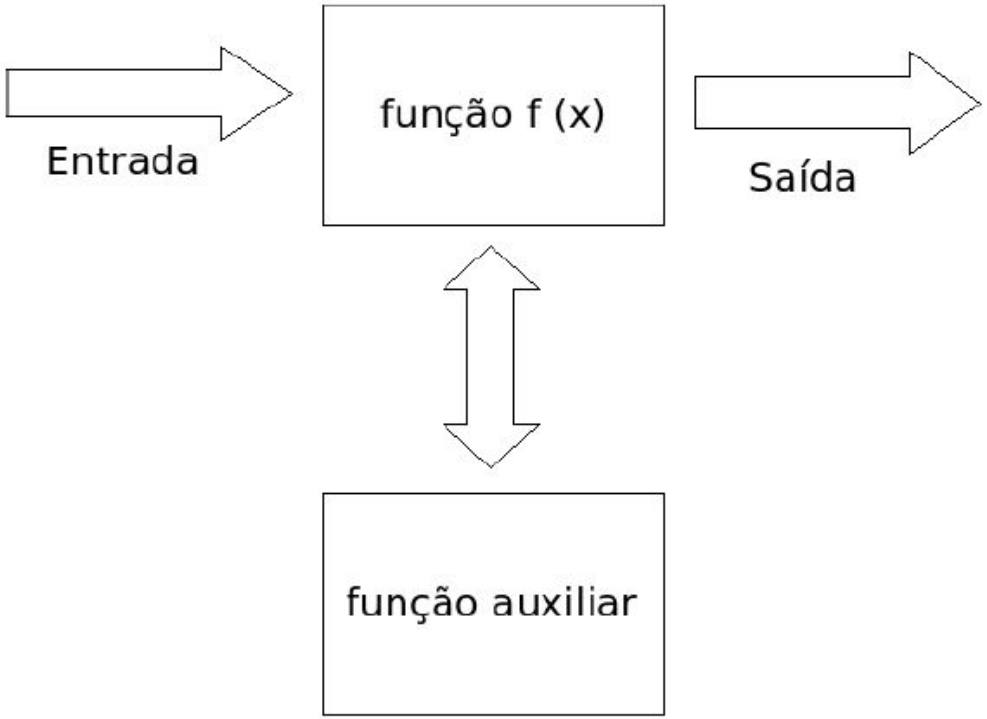
-Reutilização de código.

Funções auxiliares

Entrada: Parâmetros;

Saída: Retorno;

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



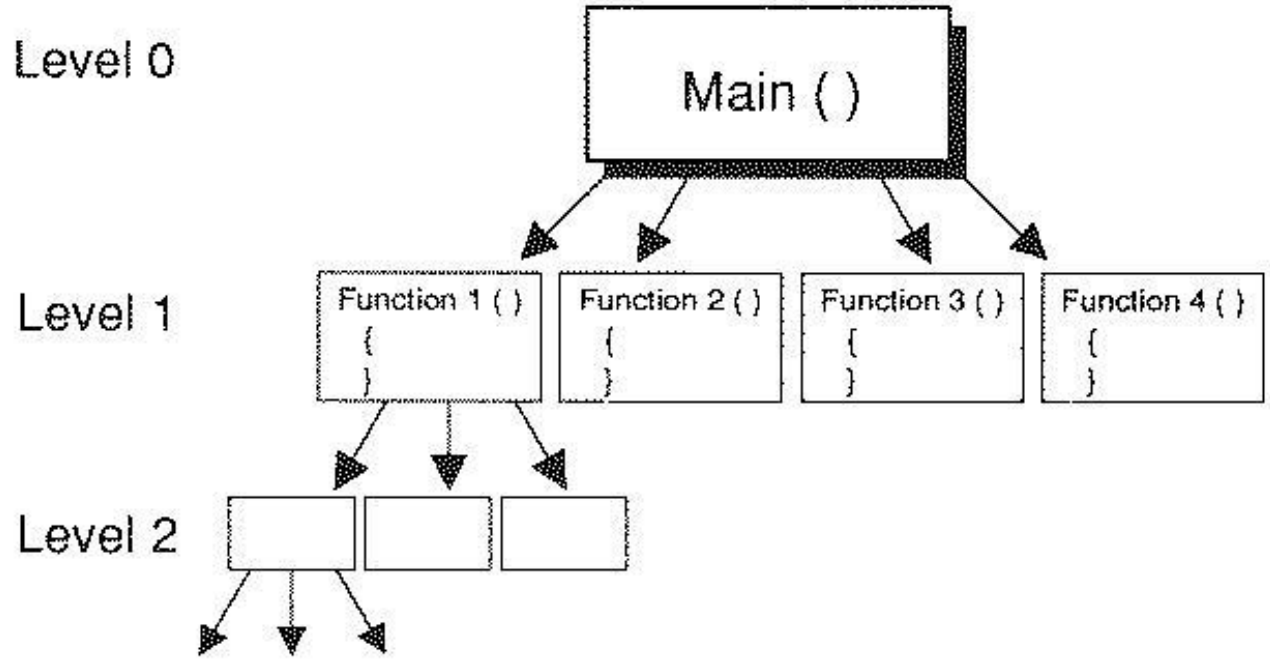
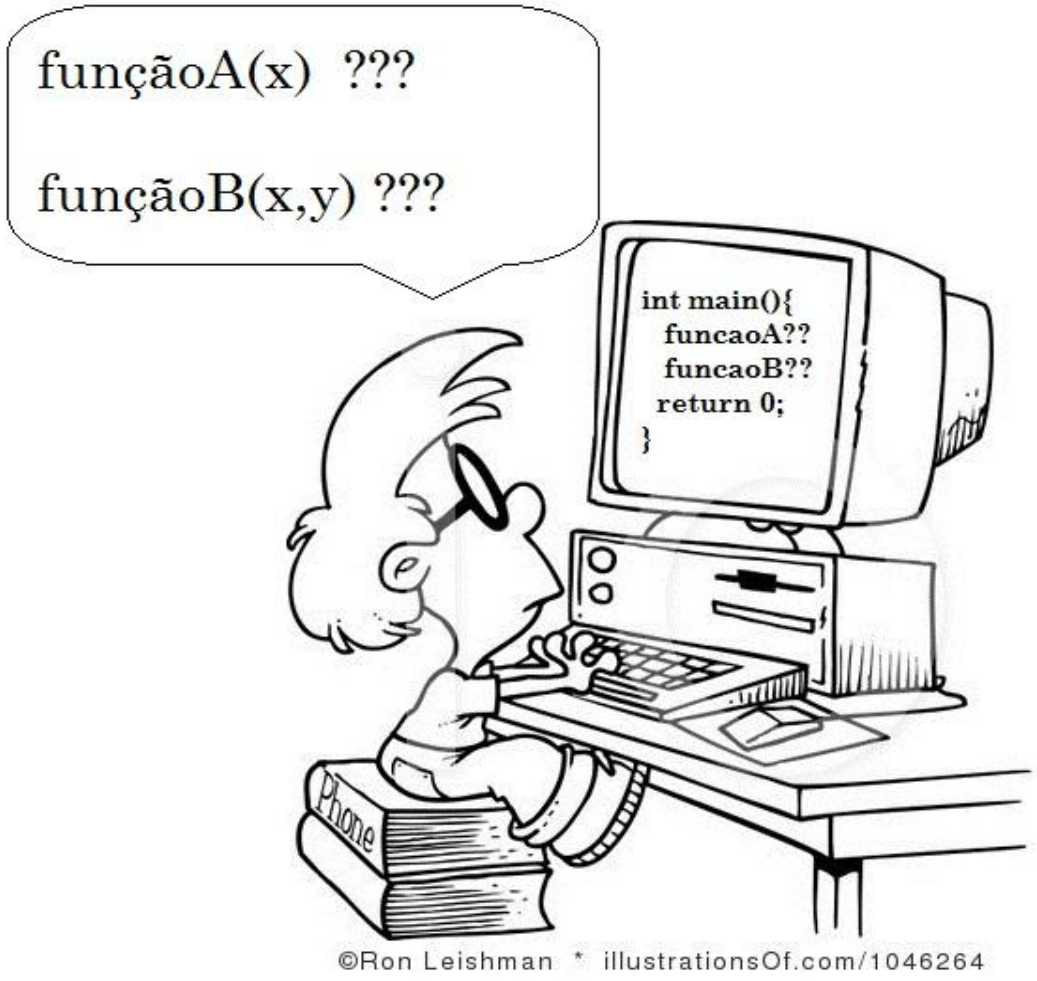


Figure 7.1. C Programs are built from functions.

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

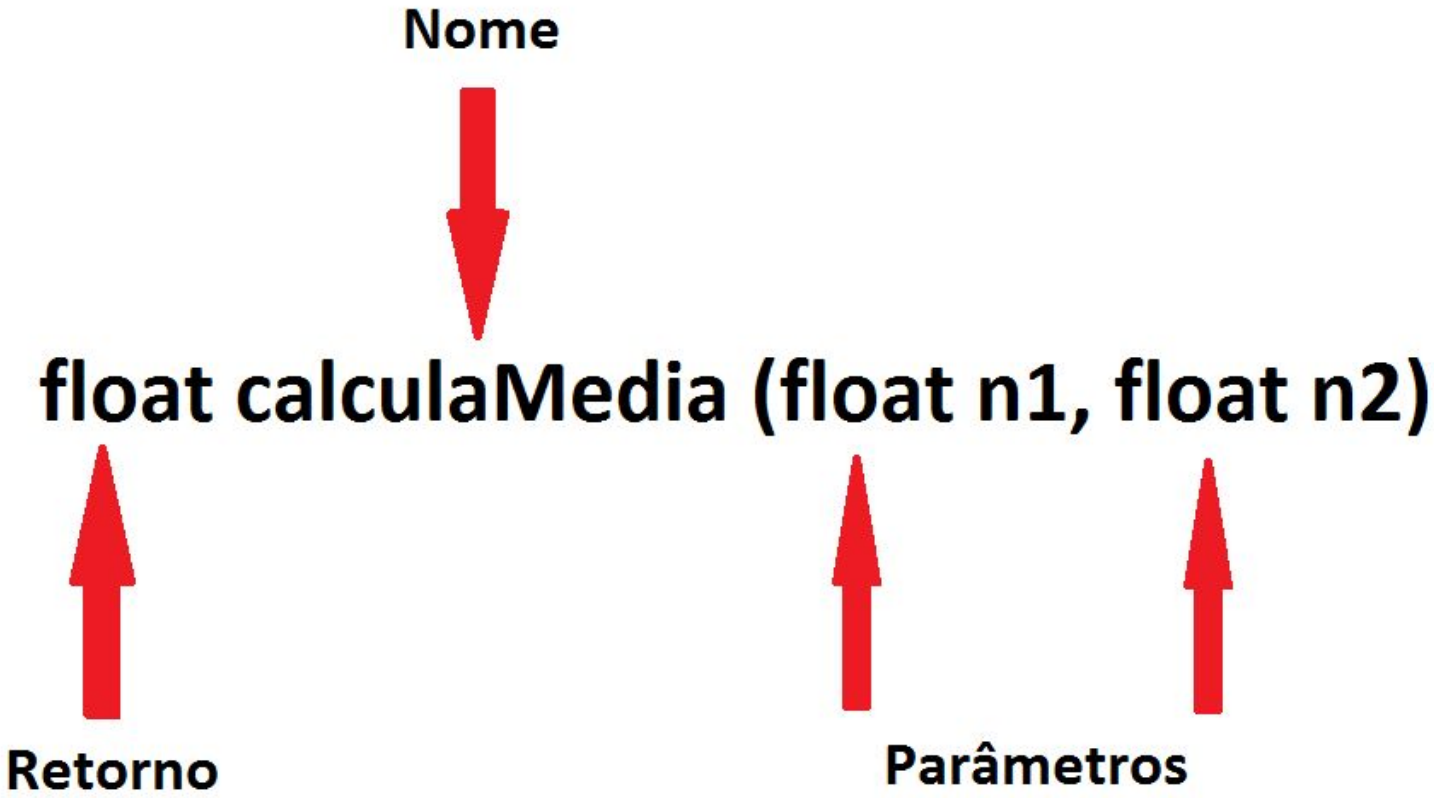
Observe a função main, ela retorna “0” (inteiro) ao fim da execução do programa e não tem parâmetro de entrada (void)

```c
#include <stdio.h>

int main(void)
{
    return 0;
}
```

Estrutura de uma função

Exemplo de cabeçalho:



Estrutura de uma função

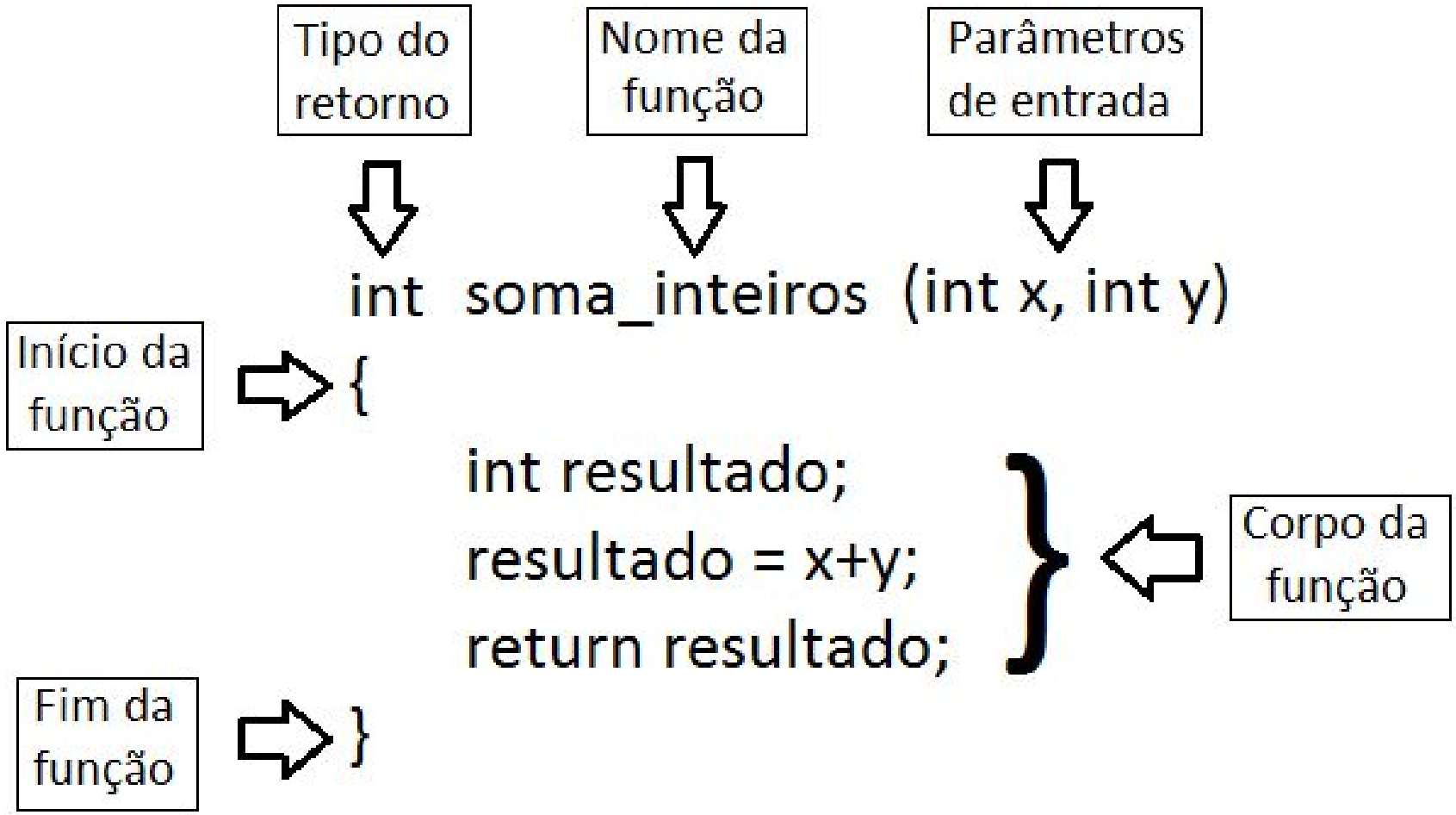
Como chamar uma função:

```
1  int main ()
2  {
3      float n1=0,n2=0,media=0;
4      scanf("%f %f",&n1,&n2);
5      media = calculaMedia(n1,n2);
6      return 0;
7  }
8
```

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  {
3      int i=0;
4      for(i=x;i<=y;i++)
5      {
6          printf("%d ", i);
7      }
8      printf("\n");
9      return;
10 }
11
```

Estrutura de uma função - exemplos

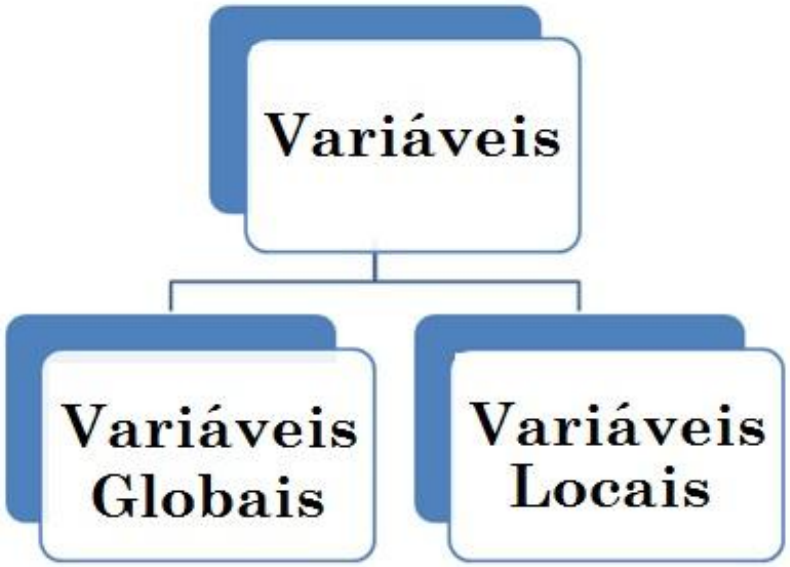
Faça uma função que retorne um inteiro lido pelo teclado.

Estrutura de uma função

```
1  int retorna_lido()
2  {
3      int a=0;
4      scanf("%d",&a);
5      return a;
6  }
7
```

Variáveis

Variáveis locais e globais:




```c
#include <stdio.h>

int a = 1; // VARIÁVEL GLOBAL

void main()
{
    int b = 2; // VARIÁVEL LOCAL

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

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.

Manipulação de Arquivos

Até agora vimos como exibir dados na tela e receber informações pelo teclado. Agora veremos como utilizar arquivos.

A manipulação de arquivos envolve 3 etapas.

1. Abrir o arquivo;
2. Ler e/ou gravar os dados desejados;
3. Fechar o arquivo.

Tipo FILE*

Para manipular arquivos precisamos definir em nosso codigo uma variavel do tipo FILE*(arquivo)

FILE* fp; // não se esqueça do asterisco

Falaremos mais sobre o significado do * em aulas futuras do Introcomp!

Comando fopen()

Para abrir um arquivo utilizaremos o comando fopen, cuja sintaxe é.

FILE* fopen(<nome_do_arquivo>,<modo_de_acesso>)

Modos de Acesso

Modo	Significado
r	Leitura. O Arquivo deve existir.
r+	Leitura e Escrita. O Arquivo deve existir.
w	Abre o arquivo somente para escrita no início do arquivo. Apagará o conteúdo do arquivo se ele já existir, criará um arquivo novo se não existir.
w+	Abre o arquivo para escrita e leitura, apagando o conteúdo pré-existente.
a	Abre o arquivo para escrita no final do arquivo. Não apaga o conteúdo pré-existente
a+	Abre o arquivo para escrita no final do arquivo e leitura.

Comando fopen()

Exemplo

```
FILE *fp;
fp = fopen (“saida.txt”, ”w”);
```

Em qual modo foi aberto o arquivo?

Comando fclose()

No final da manipulação do arquivo devemos fecha-lo com o comando fclose().

```
FILE *fp;

// Manipulação do arquivo

fclose(fp);
```

Comando fprintf()

Igual ao comando printf() porém para file.

O primeiro parâmetro a ser passado deve ser a variável do tipo FILE*

Comando fprintf()

```c
#include <stdio.h>

int main()
{
    FILE * arq;

    arq = fopen ("arquivo.txt", "w");

    fprintf(arq, "Olá Mundo!\n");

    fclose(fp);

    return 0;
}
```

Comando fprintf()

```c
#include <stdio.h>

int main()
{
   FILE * arq;
   int num;

   arq = fopen ("arquivo.txt", "w");

   printf("Digite um numero:");
   scanf("%d",&num);
   fprintf(arq, "Voce digitou o numero %d", num);

   fclose(arq);

   return 0;
}
```

Comando fscanf()

Igual ao comando scanf() porém para file.

O primeiro parâmetro a ser passado deve ser a variável do tipo FILE*

Comando fscanf()

```c
#include <stdio.h>

int main()
{
   FILE * arq;
   int num1,num2,num3;

   arq = fopen ("arquivo.txt", "r");

   fscanf(arq,"%d %d %d",&num1,&num2,&num3);
   printf("O arquivo tem os numeros %d,%d e %d\n" ,num1,num2,num3);


   fclose(arq);

   return 0;
}
```

Comando fscanf()

Retorno do fscanf: Quando o fscanf é realizado com sucesso ele retorna o número de itens lidos.

Se o final do arquivo é atingido o fscanf retorna o inteiro EOF.

Dessa forma é possível percorrer um arquivo inteiro, lendo seu conteúdo.

Comando fscanf()

```c
#include <stdio.h>

int main()
{
   FILE * arq;
   int num;

   arq = fopen ("arquivo.txt", "r");

   while(fscanf(arq,"%d",&num) != EOF){
   printf("%d\n" ,num);
  }


   fclose(arq);

   return 0;
}
```

Comando fscanf()

```c
#include <stdio.h>

int main()
{
   FILE *f;
   float n;

   f = fopen ("arquivo.txt", "r");

   while(fscanf(f,"%f,",&n) != EOF)
   {
        printf("Numero lido: %.2f\n",n);
   }


   fclose(f);

   return 0;
}

```

Abrir

```
5.5,10,9.5,7.75,9.5,9.5,10,2.75,3.5,7,8,9.5,10
```

Comando fscanf()

```
Numero lido: 5.50
Numero lido: 10.00
Numero lido: 9.50
Numero lido: 7.75
Numero lido: 9.50
Numero lido: 9.50
Numero lido: 10.00
Numero lido: 2.75
Numero lido: 3.50
Numero lido: 7.00
Numero lido: 8.00
Numero lido: 9.50
Numero lido: 10.00
```

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!!

Arquivos - Laboratório

Faça um programa que leia um número N do teclado seguido de N notas (valores reais entre 0 e 10).

Escreva em um arquivo chamado “notas.txt” as notas dos alunos, a maior nota, a menor nota e a média da turma.

Arquivos - Laboratório

Crie um arquivo .txt que contenha um número qualquer de números inteiros separados por espaço.

Faça um programa que mostre no terminal a quantidade desses números que são pares.

OBS: Utilize EOF para ler o arquivo até o final.

AULA 05

{introcomp}

MODULARIZAÇÃO