

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
MATO GROSSO DO SUL

Algoritmos II

Prof. Douglas Francisquini Toledo
Prof. Rogério Alves dos Santos Antoniassi

Aula de Hoje



- Subprogramação.

Objetivo



- Conhecer técnicas para **modularizar o problema.**



Partindo o problema em pedaços

SUBPROGRAMAÇÃO



Introdução

- **Subprogramação:** criar pequenos “programas”;
- **Objetivo:** dividir o programa em blocos;
 - **Blocos de instruções** que realizam tarefas específicas.
- Sempre há um **bloco principal**.
 - **Ex:** Em C tem-se a *main*

Introdução

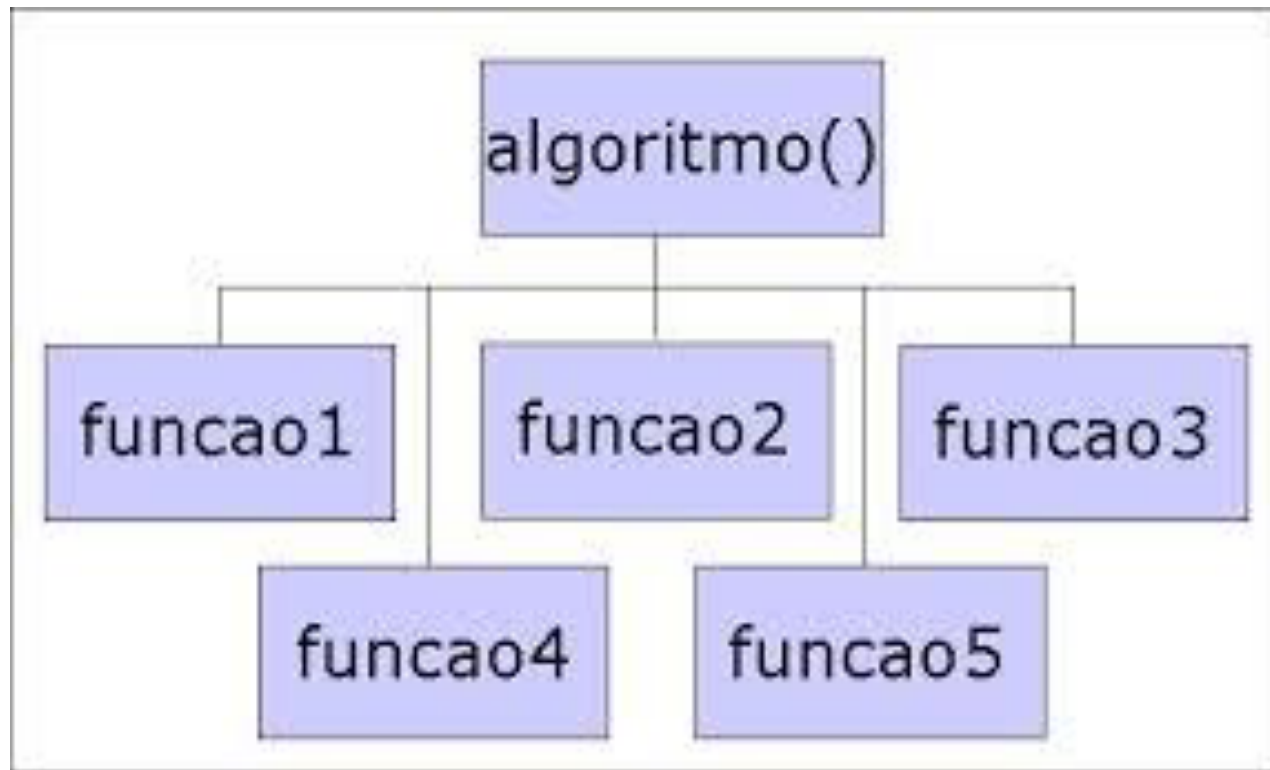


- **Vantagens:**

- Os programas tendem a **ficarem menores**;
- O código, geralmente, fica **mais organizado**;
- O problema pode ser **resolvido por partes**.
- **Entre outros.**



Introdução





Introdução

- A **Subprogramação** pode gerar desvios na maneira do programa ser executado;
 - **Não** necessariamente são **desvios condicionais**.
 - Acontecem quando o **programa principal** chama um **subprograma**.



Introdução

- **Terminologia para Subprogramação:**
 - Subprogramas;
 - Subrotinas;
 - Funções (mais utilizado);
 - Métodos;
 - **Entre outros.**

Em C



- Existem diversas **funções prontas**;
 - “Linkadas” por meio do ***#include***;
 - Ex: `printf()` e `scanf()`.
- Porém, o programador pode criar suas **próprias funções**;
 - Especificando o **tipo de retorno** e os **parâmetros**.

Em C



- As funções criadas pelo programador devem estar **fora da *main***;
 - Pois a função *main* é a **função principal**.
- Toda função deve ser chamada **dentro da *main*** para que **seja executada**.
 - Seja criada ou pronta em alguma biblioteca.

Em C



- **Formato:**

Um tipo de dado qualquer, tais como, **int**, **void** e **char**.

<tipo_retorno> <nome_função>(parâmetros)

{

comando1;

O nome da função segue as mesmas regras dos **nomes das variáveis**

Valores que serão passados por quem chama a função.

return <valor_tipo_retorno>;

}

A função deve retornar um valor do mesmo tipo que ela indicou em seu cabeçalho.

Em C



- Exemplo:

```
void imprime (int valor)
{
    printf("O valor eh %d", valor);

    return;
}
```

Em C



- Exemplo:

```
int soma (int valor_1, int valor_2)
{
    int resultado;

    resultado = valor1 + valor_2;

    return resultado;
}
```

Em C



- Exemplo:

```
float media (float valor_1, float valor_2)
{
    float resultado;

    resultado = ( valor_1 + valor_2 ) / 2;

    return resultado;
}
```

Em C



- Exemplo:

```
char numeroPar (int numero)
{
    int resto;
    char resposta;

    resto = numero % 2;

    if (resto == 0)
        resposta = 'S';
    else
        resposta = 'N';

    return resposta;
}
```


Em C



- Exemplo:

```
int main (void)
{
    ...
    imprime (10);
    s = soma (5, 7);
    m = media (6.5, 8.75);
    p = numeroPar (13);
    ...
}
```

Em C



- O **uso dos delimitadores** (“{” e “}”) é obrigatório;
- Podemos colocar **qualquer coisa** dentro da função;
 - Toda **função** ou **comando** que seja válido na em C;
 - Inclusive, **chamar uma outra função** que fizemos;
 - **Declarar novas variáveis** dentro das funções;
 - **Ex:** printf, scanf, if e else e operações aritméticas.

Discussão



- Dúvidas?





Exercícios

1. Faça uma função que retorne 1 se o número digitado for positivo e 0 se o número digitado for negativo.
2. Faça uma função que receba dois números positivos por parâmetro e retorne a soma dos N números inteiros existentes entre eles.



Exercícios

3. Dada uma sequência de n números, calcule a soma de todos os números primos e exiba o resultado.

Para tanto crie duas funções: uma para verificar se o número é primo e outra para realizar a soma dos números primos.



Próximas aulas, recados e discussões

FECHAMENTO



Conclusão

- Para que uma função funcione ela precisa ser chamada **pela função *main***;
 - A função *main* funciona como um **programa principal**.
- Os parâmetros podem ser de **tipos diferentes**;
 - **Ex:** Um parâmetro do tipo *float* e outro do tipo *int*.

Curiosidades



- **Variáveis globais vs variáveis locais:**
 - Globais para o programa todo;
 - Locais apenas para a função em questão;
 - Inclusive, existem variáveis locais na ***main***.



Bibliografia Básica

- ASCENCIO, Ana F. G., CAMPOS, Edilene V. **Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ (padrão ANSI) e Java.** 3. ed. São Paulo: Pearson, 2012.
- CORMEN, Thomas et al. **Algoritmos: teoria e prática.** Rio de Janeiro: Câmpus, 2002.
- SCHILDT, Herbert. **C completo e total.** 3. ed. São Paulo: MakronBooks, 1997.