Roteiro de Estudos

# Algoritmos e Técnicas de Programação

Este roteiro orientará a sua aprendizagem por meio da leitura de livros e artigos que cabem na sua rotina de estudos. Experimente esse recurso e aumente a sua habilidade de relacionar a teoria à prática profissional.
No seu caminho de aprendizagem, você encontrará os seguintes tópicos:

✓ Texto de apresentação de cada leitura indicada;
✓ Links para acesso às referências bibliográficas.

É importante ressaltar: o seu esforço individual é fundamental para a sua aprendizagem, mas você não estará sozinho nessa!

# UNIDADE 4

## FUNÇÕES E RECURSIVIDADE

### Procedimentos e Funções

Dividir para conquistar (“Divide et impera” ou “Divide et Vinces”) é um clássico nas estratégias de guerra para enfraquecer e subjugar os povos. O termo, embora já fosse conhecida na Antiguidade, foi cunhado por Júlio César em seu livro “De Bello Gallico” (Guerra das Gálias), que explicou como a vitória romana na guerra gaulesa era essencialmente uma política de “dividir” seus inimigos, aliar com tribos individuais durante suas disputas com adversários locais. No mundo computacional o pensamento é o mesmo.

A ideia de criar programas com blocos de funcionalidades vem de uma técnica de projeto de algoritmos chamada dividir para conquistar (MANZANO; MATOS; LOURENÇO, 2015). Isso evita que os códigos façam retrabalho e fiquem mais curtos, para isso, as linguagens oferecem recurso para criar funções e procedures. Uma função é definida como um trecho de código escrito para solucionar um subproblema (SOFFNER, 2013).

Imagine que uma determinada máquina cada vez que recebe 10 valores, tem que realizar a média e encaminhar para algum lugar pré-determinado, esse ciclo é repetido 30 vezes ao dia, se não existisse a possibilidade da criação de funções, o código teria que ter 30 vezes o cálculo da média, imagine o tamanho do código só para fazer esse retrabalho. Para fazermos programas grandes e complexos é necessário de construí-los bloco por bloco. Uma função é um bloco que tem por objetivo calcular e retornar ao algoritmo, essa função pode ser chamada a qualquer momento e quantas vezes forem necessárias. Outra característica da utilização de funções é que

elas “quebram” a linearidade de execução, pois a execução pode “dar saltos” quando uma função é invocada (SOFFNER, 2013).

Vamos analisar o formato de uma função:

```
#include<stdio.h>

int media(a,b,c)

{ return (a+b+c)/3; }

int main()

{

int resultado = 0;

resultado = media(12,45,25);

printf("O resultado da funcao e = %d",resultado);

return 0;

}
```

Foi criada a função media(), como a função retorna um valor, o tipo de dado deve ser referenciado, no caso do exemplo, inteiro (int media()). Dentro do programa principal, é realizada a chamada da função e quando esta retornar o valor a variável resultado recebe. Os valores que estão entre parênteses na chamada da função, chamamos de parâmetros, que serão armazenados respectivamente em a,b e c, para que a média possa ser calculada. Então, é possível realizar quantas chamadas forem necessárias.

Como no caso do robô que recebe vários dados por dia e tem que encaminhar para processamento, na programação irá aparecer vários chamados:

```
resultado = media(12,45,25);

...

resultado = media(14,46,79);

...

resultado = media(45,78,32);

...

Etc.
```

# Referências e Link do material na Biblioteca Virtual e artigo

SOFFNER, R. **Algoritmos e programação em linguagem C**. São Paulo: Saraiva, 2013.

MANZANO, J. A. N. G. **Linguagem C: acompanhada de uma xícara de café**. São Paulo: Érica, 2015. 480 p.

SCHEFFER, Vanessa Cadan. ARTERO, Marcio Aparecido. **Algoritmos e técnicas de programação** Londrina: Editora e Distribuidora Educacional S.A., 2018.

# Escopo e Passagem de Parâmetros

Caro aluno, com o recurso das funções, os programas podem ser separados em blocos, isso facilita muito a visualização e o tamanho do código final. Os blocos devem "conversar" entre si, ou seja, os programas chamadores devem ter condições de enviar os dados que devem ser processados, esses dados que são enviados são chamados de parâmetros.

Existem algumas regras em relação a utilização de uma variável em um programa, isso Chamamos de escopo de variável. Essas variáveis podem ser divididas em três tipos: variáveis locais, parâmetros formais e variáveis globais.

As variáveis locais, são aquelas declaradas dentro do bloco de uma função, não podem ser usadas ou modificadas por outras funções e somente existem enquanto a função onde foi declarada estiver sendo executada.

Os parâmetros formais de uma função também são variáveis locais da função.

Existem também as variáveis globais que são acessíveis em qualquer parte do programa, ou seja, podem ser usadas e modificadas por todas as outras funções, são declaradas fora de todos os blocos de funções e existem durante toda a execução do programa.

Os parâmetros da função na sua declaração são chamados **parâmetros formais**. Na chamada da função os parâmetros são chamados **parâmetros atuais/reais**.

Vejamos o código:

```
#include<stdio.h>

Int multiplica (int N1, int N2)

{    int resultado;
```

```
    resultado = N1 * N2;
    printf ("Resultado = %d\n", resultado);  }
main ()
{    multiplica (3, 5);
     multiplica (6, 5);
     multiplica (2, 8);  }
```

Os parâmetros são passados para uma função de acordo com a sua posição. Ou seja, o primeiro parâmetro atual (da chamada) define o valor o primeiro parâmetro formal (na definição da função, o segundo parâmetro atual define o valor do segundo parâmetro formal e assim por diante. Podemos definir como sendo o escopo de uma variável "a relação de alcance que se tem com o local onde certo recurso se encontra definido, de modo que possa ser "enxergado" pelas várias partes do código de um programa" (MANZANO, 2015, p. 288).

Observe que a função main() e a função multiplica() possuem variáveis com o mesmo nome "resultado". Para a Linguagem C, isso não gera confusão, pois, apesar do mesmo nome, elas são variáveis diferentes. Para a função multiplica, o parâmetro resultado funciona como uma variável local (ou seja, válida dentro da função).

## Referências e Link do material na Biblioteca Virtual e artigo

MANZANO, J. A. N. G. **Linguagem C: acompanhada de uma xícara de café.** São Paulo: Érica, 2015.

## Recursividade

Caro aluno, normalmente uma função é executada quando algum procedimento é chamado, porém, existem casos onde a própria função chama a sim mesma, parece estranho, mas existe e funciona, chamamos esse procedimento de recursividade. Nessa técnica, uma função é dita recursiva quando ela chama a si própria ou, nas palavras de Soffener,: "recursividade é a possibilidade de uma função chamar a si mesma." (SOFFENER, 2017, p. 107).

Embora a sintaxe da função recursiva seja similar as não recursivas, o funcionamento de ambas é bastante distinto e o mau uso dessa técnica pode acarretar em uso indevido de memória, muitas vezes chegando a travar a aplicação e o sistema (MANZANO, 2015).

Existem algumas desvantagens no uso da recursividade, quando o loop recursivo é muito grande, uma grande quantidade de memória é consumida, pois cada chamada recursiva aloca memória para os parâmetros. Realizar análises criteriosas do sistema é importante, pois, em muitos casos uma solução iterativa gasta menos memória, e torna-se mais eficiente em termos de performance do que usar recursão.

O seguinte código ilustra o modelo de recursividade.

```
#include <stdio.h>

int fatorial(int n)
{
   if(n == 1)
      return 1;
   else
      return ( n * fatorial(n-1) );
}

int main()
{
   int n;
   printf("Digite um inteiro positivo: ");
   scanf("%d", &n);

   printf("%d! = %d\n", n, fatorial(n));
}
```

recursividade

Chamada

O resultado será exibido assim:

```
Digite um inteiro positivo: 5
5! = 120


...Program finished with exit code 0
```

*Elaborado pelo autor. 1*

- O cálculo realizado para o fatorial é o seguinte: $5! = 5 * 4 * 3 * 2 * 1 = 120$.

A função main() irá chamar a função fatorial, somente quando o valor da variável n for igual a 1, a função retorna para a chamadora, enquanto isso, a própria função fatorial tem a função de chamar a si mesma.

Toda função recursiva tem que possuir um critério de parada. A instância da função que atenderá a esse critério é chamada de caso base.

## Referências e Link do material na Biblioteca Virtual e artigo

SOFFNER, R. Algoritmos e programação em linguagem C. São Paulo: Saraiva, 2013.

SCHEFFER, Vanessa Cadan. ARTERO, Marcio Aparecido. **Algoritmos e técnicas de programação** Londrina: Editora e Distribuidora Educacional S.A., 2018.