FUNÇÕES

A maioria dos programas de computador são maiores do que os que desenvolvemos ao longo do estudo de lógica de programação. A melhor maneira de desenvolver e manter um programa grande é construí-lo a partir de partes menores, que são conhecidas como **FUNÇÕES**.

Além das funções construídas pelos programadores, a biblioteca do C++ oferece uma rica coleção de funções já prontas para cálculos matemáticos, operações com strings, entradas e saídas de dados, entre outras. Vejamos um exemplo da função pow da biblioteca <cmath> ou <math.h>

```
int base = 3, exponente = 6;
int novonumero;
novonumero = pow(base, expoente);
```

Quando na última linha de código utilizamos a função **pow**, estamos nos referindo a função **pow** da biblioteca **<math.h> ou <cmath>**. A função **pow** eleva um número a uma determinada potência. Estas funções presentes na biblioteca facilitam o trabalho do programador porque estas funções oferecem muitos dos recursos de que os programadores precisam. Evite reinventar a roda, sempre que possível e se existir, utilize as funções da biblioteca ao invés de escrever funções novas.

As funções permitem ao programador modularizar um programa. Todas as variáveis declaradas em definições de função são variáveis locais, ou seja, elas só são conhecidas na função na qual são definidas. A maioria das funções possui uma lista de parâmetros que são responsáveis por prover os meios para transferir informações entre funções. Os parâmetros de uma função também são variáveis locais. O formato da chamada da função inclui a especificação do nome colocando os argumentos dos parâmetros entre parênteses.

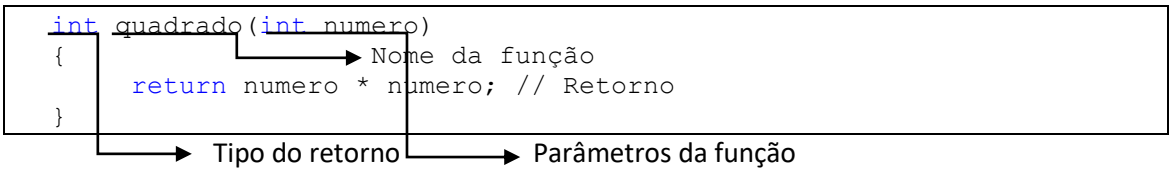
Como já sabemos, uma função é um bloco de instruções que é executado quando é chamada de algum outro ponto do programa. Seu formato é o seguinte:

```
tipo nome ( parametro1, parametro 2, ...) conteúdo
```

- **tipo:** é o tipo de dados que a função retorna.

- **nome:** é o nome pelo qual será possível chamar a função.
- **parâmetro:** (podem ser especificados quantos você quiser). Cada argumento consiste em um tipo de dados seguido pelo seu identificador, como na declaração de uma variável (por exemplo, int x) e que funciona dentro da função como qualquer outra variável. Eles permitem a passagem de parâmetros para a função quando é chamada. Os parâmetros diferentes são separados por vírgulas.
- **conteúdo:** é o corpo da função. Pode ser uma única instrução ou um bloco de instruções. No último caso, precisa ser delimitado por chaves {}.

Considere o exemplo abaixo onde temos a definição da função **quadrado**. A função retorna um inteiro e possui um parâmetro onde é informado o número que será elevado ao quadrado.



Vamos agora lembrar da sintaxe de uma declaração de função: **tipo nome (parametro1, parametro2 ...) conteúdo**. Estudamos que é obrigatório que a declaração comece com um tipo, que é o tipo de dados que será retornado pela função com a instrução **return**. Acontece que no C++ podemos escrever funções sem retorno de valor.

Imagine que necessitamos fazer uma função somente para exibir uma mensagem na tela. Nós não precisamos que retorne nenhum valor, e mais ainda, não precisamos receber nenhum parâmetro. Para casos assim podemos utilizar uma função com retorno **void**.

Mais um exemplo:

```
//define uma função para imprimir algo na tela
void imprimirNaTela()
{
    cout << "FUNÇÃO DO TIPO VOID" << endl;
}
```

Exercícios

QUESTÃO 01. O máximo divisor comum de dois inteiros é o maior inteiro que divde cada um dos dois números. Escreva uma função MDC que retorna o máximo divisor comum de dois inteiros.

QUESTÃO 02. Escreva uma função que receba a média de um aluno e retorne 4, se ela estiver no intervalo 90-100, 3 se a média estiver no intervalo 80-89, 2 se a média estiver no intervalo 70-79, 1 se a média estiver no intervalo 60-69, e 0 se a média for menor que 60.

QUESTÃO 03. Faça uma função que recebe, por parâmetro, a altura (alt) e o sexo de uma pessoa e retorna o seu peso ideal. Para homens, calcular o peso ideal usando a fórmula peso ideal = 72.7 x alt - 58 e, para mulheres, peso ideal = 62.1 x alt - 44.7.

QUESTÃO 04. Escreva uma função que recebes 3 valores reais X, Y e Z e que verifique se esses valores podem ser os comprimentos dos lados de um triângulo e, neste caso, retornar qual o tipo de triângulo formado. Para que X, Y e Z formem um triângulo é necessário que a seguinte propriedade seja satisfeita: o comprimento de cada lado de um triângulo é menor do que a soma do comprimento dos outros dois lados. Uma função deve identificar o tipo de triângulo formado observando as seguintes definições:

- Triângulo Equilátero: os comprimentos dos 3 lados são iguais.
- Triângulo Isósceles: os comprimentos de 2 lados são iguais.
- Triângulo Escaleno: os comprimentos dos 3 lados são diferentes.

QUESTÃO 05. Faça uma função que recebe, por parâmetro, um valor N e calcula e escreve a taboada de 1 até N. Mostre a tabuada na seguinte forma abaixo:

```
1 x N = N

2 x N = 2N

...

N x N = N2
```

QUESTÃO 06 (DESAFIO). Dados dois strings (um contendo uma frase e outro contendo uma palavra), determine o número de vezes que a palavra ocorre na frase.

Exemplo:

Para a palavra ANA e a frase :
ANA E MARI**ANA** GOSTAM DE B**ANANA** (2)

Temos que a palavra ocorre 4 vezes na frase.

QUESTÃO 07. Faça uma função que obtenha do teclado o valor n e imprima na tela os n primeiros termos de uma seqüência de Fibonacci.
Observação: Uma seqüência é dita de Fibonacci quando os dois primeiros termos forem 0 e 1 e todos os termos consecutivos tenham valor igual à soma dos dois termos anteriores.

Exemplo: 0 – 1 – 1 – 2 – 3 – 5 – 8 – 13 – 21 – 34 – 55 .