

## Função



## Revisão de Funções

## Definição (função secundária)

 É um conjunto de comandos agrupados em um bloco, separado do programa principal, que pode ser chamada a qualquer momento da execução do programa de forma a cumprir um determinado objetivo.

### Vantagens

 As principais vantagens em se utilizar funções se devem ao fato de que elas ajudam a fragmentar o código em partes menores - mais fáceis de se compreender e ainda por cima podem ser utilizadas mais de uma vez no mesmo programa, poupando preciosos minutos de programação e inúmeras linhas de código



## **Estrutura**

#### Tipo da Função:

• É o tipo de dado retornado pela função ao programa ou função principal, que pode ser numérico, literal ou lógico (int, float, double, char, bool...).

#### Tipo de Parâmetros:

• É o tipo de dado de cada parâmetro a ser passado pelo programa ou função principal (main).



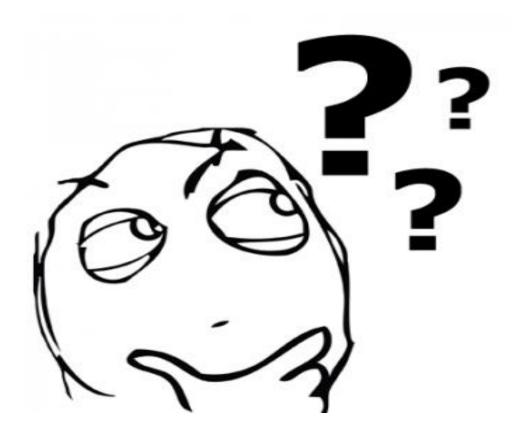
#### Lista de Parâmetros:

 As variáveis que passarão informações à função são chamadas de parâmetros e a função deve declarar esses parâmetros entre parênteses, no cabeçalho da definição da função. Esses parâmetros podem ser utilizados livremente no corpo da função.

#### **Return:**

termina a execução da função e retorna o controle para a instrução seguinte no programa ou função principal (main), após a chamada da função. O comando return pode retornar somente um único valor para o programa chamador. A sintaxe do return tem 4 formas: return constante (ex: retorne o valor -1), return variável (retorne o conteúdo da variável soma), return expressão (retorne o resultado da expressão aritmética (F-32)\*5/9) ou return (usado para encerrar a função usado em funções void)





Alguma dúvida?



## Tipos de funções

#### Com base nos parâmetros:

- Sem passagem de parâmetro.
- Com passagem de parâmetro por cópia.
- Com passagem de parâmetros por referência.

#### Com base nos tipos de função:

- Com retorno.
- Sem retorno



# Passagem de parâmetros por Referência.



**FOCO DE HOJE** 



#### **Ponteiros:**

 Um ponteiro é simplesmente uma variável que aponta para o endereço de memória de outra variável. É possível criar um ponteiro para qualquer tipo de variável em C/C++.



```
int soma(int *a, int *b) {
  return *a + *b;
  }
```

#### Endereço de memória

 Ao se declarar uma variável, um espaço na memória do computador é reservado para armazená-lá. Para acessar o endereço de memória em C++ basta adicionar o símbolo '&' antes da variável a qual se deseja obter o endereço.



#### Definição

 Ao se passar um valor por referência é necessário trabalhar com ponteiros e endereços, e se deve ao fato de que os parâmetros da função se tornam ponteiros para a variável que foi passada por parâmetro. Sendo assim, é necessário ao se chamar a função passar o endereço de cada variável;

#### **Exemplo:**

```
#include <iostream>
 2 using namespace std;
    int dobro(int *a, int *b)
 4 ∃ {
 5
       int x;
       *b = 2;
        x = *a * *b;
         return x;
   int main()
11 ⊟ {
12
        int a = 2, b = 3, c;
13
        c = dobro(\&a, \&b);
        cout << "A: " << a << endl; // Note que o valor da variável b foi
14
15
        cout << "B: " << b << endl; // alterado na função dobro(), e essa
        cout << "C: " << c << endl; // alteração será valida também na função principal
16
17
         return 0;
18 L }
```





## Mão na massa



## Exercícios

1) Sabendo que existe uma variável inteiro A e uma variável inteiro B, use a passagem de parâmetros por referência para criar uma função que retorne o maior desses vetores.



2) Um aluno do INATEL fez a av1 e a av2 de uma determinada matéria. Sabendo que se a média desse aluno for maior que 60 ele passará direto, se for menor que 60 e maior que 30 ele ficará de NP3 e que se sua nota for inferior a 30 ele irá reprovar direto. Faça um código que use uma função com passagem por referência para identificar se esse aluno "passou", "pegou Np3" ou se "reprovou".