

Declaração de Função

Há situações em que um trecho do programa é executado mais de uma vez em locais diferentes do código. Isso deixa o código com trechos repetidos que podem ser separados do principal e serem executados quando forem necessários.

Para esses trechos são declaradas funções responsáveis pela execução desse código. Na declaração deve ser especificados os parâmetros, se forem necessários. E caso retorne algum valor, também deve ser especificado o tipo de retorno.

```
funcao tipo nome_função (tipo parametro1, tipo paramentroN)
{
    retorne conteúdo, somente se não retornar vazio
}
```

A declaração dos parâmetros de uma função segue o mesmo princípio de variáveis não inicializadas, já que são utilizados para receber valores que podem ser diferentes cada vez que a função for executada.

VOCÊ SABIA?

Em Portugol Studio, o nome da função segue as regras para nome de variável, ou seja, utilizar letras e números, mas não iniciar com números e o único caracter especial que é permitido é o *underline* (_).

As funções podem ser declaradas em qualquer local dentro do programa, conforme observado na Figura 1. Mas nunca declare uma função dentro de outra função.

Quando as funções retornam algum valor, deve-se colocar o comando

“**retorne**” e em seguida o conteúdo a ser retornado de acordo com o tipo de retorno declarado na função. Normalmente, esse comando é colocado no final da função, pois encontrá-lo a função é encerrada.

Figura 1: Exemplos de declaração de função

```
programa
{
    funcao inicio()
    {
    }

    //Declaração de função sem retorno e sem parâmetros
    funcao exibelinha()
    {
    }

    //Declaração de função com retorno do tipo cadeia e sem parâmetros
    funcao cadeia horario()
    {
        retorne "18:25"
    }

    //Declaração de função sem retorno e com parâmetro do tipo inteiro
    funcao exibeAsteriscos(inteiro qtdeAsterisco)
    {
    }

    //Declaração de função com retorno do tipo real e com parâmetros do tipo real
    funcao real soma(real numero1, real numero2)
    {
        retorne 2.0
    }
}
```

Fonte: Elaborado pela autora

Funções

Função em linguagem de programação é um procedimento ou uma rotina que executa parte de um programa maior ou até mesmo uma situação específica. Também utilizada para dividir um programa grande em várias partes.

O uso de funções traz muitas vantagens, como:

- Evita código duplicado no mesmo programa;
- A separação de situações grandes em partes pequenas;

- Melhor organização do programa;
- Possibilidade de utilizar a mesma função em mais de um programa.

Para o uso de funções, é preciso:

- Declarar a função definindo tipo de retorno e parâmetros, se necessários;
- Programar o código, no corpo da função, para resolver o propósito da função;
- Fornecer um valor de retorno, se a função exigir, para ser utilizado após a invocação da função.

Após criar a função, ela precisa ser utilizada em algum momento e para isso é necessário invocar ou chamar a função passando os parâmetros necessários de acordo com o tipo definido nela.

VOCÊ SABIA?

O Portugol Studio já possui uma função padrão denominada "**funcao inicio()**". Esta função é obrigatória em todo programa feito nesta linguagem de programação. Sem ela, o programa não é executado e é nessa função que são invocadas todas as outras funções do programa. Porém, uma função também pode invocar outra função em execução.

A Figura 2 demonstra exemplos de funções declaradas através de formatos diferentes e suas respectivas invocações tanto na função "**inicio()**", quanto na função "soma".

Figura 2: Exemplo de uso de funções

```
programa
{
    inclui biblioteca Calendario

    funcao inicio()
    {
        //Invocação de todas as funções declaradas neste programa
        exhibeLinha()
        escreva(horario())
        exhibeAsteriscos(5)
        escreva(soma(1.5,2.3))
    }

    //Declaração de função sem retorno e sem parâmetros
    funcao exhibeLinha()
    {
        escreva("\n-----\n")
    }

    //Declaração de função com retorno do tipo cadeia e sem parâmetros
    funcao cadeia horario()
    {
        retorne Calendario.hora_atual(falso)+":"+Calendario.minuto_atual()
    }

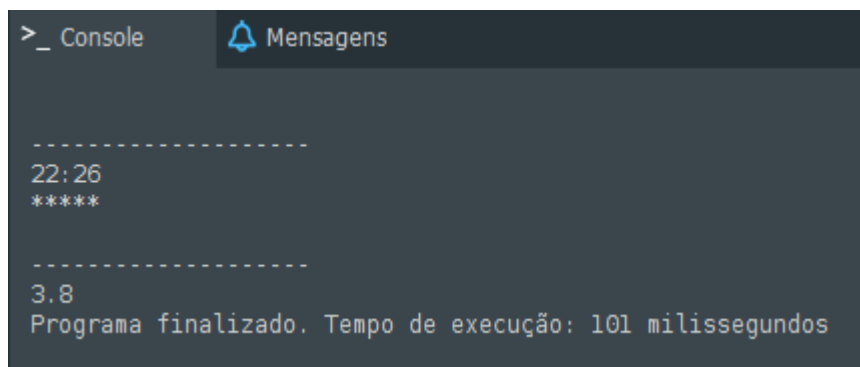
    //Declaração de função sem retorno e com parâmetro do tipo inteiro
    funcao exhibeAsteriscos(inteiro qtdeAsterisco)
    {
        escreva("\n")
        para (inteiro i=1; i<=qtdeAsterisco; i++)
        {
            escreva("*")
        }
        escreva("\n")
    }

    //Declaração de função com retorno do tipo real e com parâmetros do tipo real
    funcao real soma(real numero1, real numero2)
    {
        exhibeLinha() //Invocação de função
        retorne numero1 + numero2
    }
}
```

Fonte: Elaborado pela autora

Na Figura 3 é apresentado o resultado da execução do programa na Console.

Figura 3: Console do programa que apresenta exemplo de funções



```
> _ Console  Mensagens

-----
22:26
*****
-----
3.8
Programa finalizado. Tempo de execução: 101 milissegundos
```

Fonte: portugol-studio.jar, 2022

Como cada função é um código próprio, as variáveis declaradas dentro do corpo da função pertencem somente à esta função, ou seja, seu uso só está disponível dentro da função em que foi declarada, pois é uma variável local que pertence somente ao escopo da função de sua declaração. Portanto, é possível existir variáveis com mesmo nome em funções diferentes.

Já as variáveis que são declaradas fora de qualquer função, denominadas de variáveis globais podem ser utilizadas por qualquer função e todas as funções visualizam seu valor atual de memória.

Cada vez que uma função é invocada, os valores de suas variáveis são inicializadas, pois é uma nova execução desse código do programa. Somente a função "**inicio()**" não perde os valores de suas variáveis, visto que é a função principal que está sempre em execução.

Então, quando for necessário ter um valor disponível em qualquer parte do programa, a declaração da variável deverá ser feita fora do corpo de qualquer função e quando for um conteúdo somente para uma determinada função, a sua declaração deve ser feita dentro da função que a utiliza.

A Figura 4 apresenta um exemplo de uso de função com variáveis globais e locais.

Figura 4: Exemplo de função com variáveis globais e locais

```
programa
{
    // Declaração de variável Global
    inteiro soma=0

    funcao inicio()
    {
        soma = 1
        escreva(soma+"\n")
        escreva(adicao(2,3)+"\n")

        // A variável "resposta" não foi declarada neste escopo
        escreva(resposta)
    }

    funcao inteiro adicao(inteiro numero1, inteiro numero2)
    {
        // Declaração de variável Local
        inteiro resposta

        // A variável "soma" está sendo utilizada no cálculo
        resposta = soma + numero1 + numero2

        retorne resposta
    }
}
```

Fonte: Elaborado pela autora

VOCÊ SABIA?

Função que retorna um valor permite que ele seja utilizado de qualquer forma no programa após sua invocação. Assim, a função que não exibe nada no Console e retorna um valor pode ser utilizada mais vezes em qualquer contexto do programa, independente de como será tratado o valor retornado.

Passagens por Valor

Na criação de uma função pode ser importante que ela execute seu código com valores diferentes cada vez que é invocada o que torna a função mais útil na programação. Esses valores são recebidos pela função em forma de parâmetros, passados quando a função é invocada. Neste caso, o que ocorre é somente a passagem por valor, já que os parâmetros definidos na função não precisam ter o mesmo nome das variáveis para chamar a função.

No exemplo da Figura 5, na função principal são definidas duas variáveis denominadas de **n1** e **n2**. Quando a função **soma** é chamada, essas variáveis são informadas para que seus valores sejam passados para a função. Já na criação da função são definidos os nomes dos parâmetros para receber os valores como **valor1** e **valor2**, ou seja, as nomes utilizados para a passagem por valor entre a invocação da função e a própria função podem ser diferentes, pois dentro da função são utilizados somente os nomes definidos nos parâmetros.

Figura 5: Exemplo de função com passagens por valor

```
programa
{
    funcao inicio()
    {
        inteiro n1, n2

        escreva("Digite o 1º número: ")
        leia(n1)
        escreva("Digite o 2º número: ")
        leia(n2)

        soma(n1,n2) // chama a função
    }

    funcao soma(inteiro valor1, inteiro valor2)
    {
        inteiro resultado = valor1 + valor2
        escreva("Soma = ", resultado)
    }
}
```

Fonte: Elaborado pela autora

Então, na construção de função com passagem por parâmetros na forma de valor, o que é passado no exemplo da Figura 5, não são as

variáveis **n1** e **n2**, e sim são os conteúdos de cada uma dessas variáveis.

A passagem por valor também pode ser por meio de constantes ou valores fixos na invocação da função (Figura 6).

Figura 6: Exemplo de função com passagens por valor com constante e valor fixo

```
programa
{
    funcao inicio()
    {
        const inteiro NUMERO10 = 10
        inteiro n1, n2

        escreva("Digite o 1º número: ")
        leia(n1)
        escreva("Digite o 2º número: ")
        leia(n2)

        // chama a função somente com variáveis
        soma(n1,n2)

        // chama a função com a constante
        soma(n1,NUMERO10)

        // chama a função com valor fixado na sua invocação
        soma(5,n2)
    }

    funcao soma(inteiro valor1, inteiro valor2)
    {
        inteiro resultado = valor1 + valor2
        escreva("\nSoma = ", resultado)
    }
}
```

Fonte: Elaborado pela autora

Passagens por Referências

Quando a função simplesmente recebe valores através de parâmetros, a variável utilizada na invocação da função não tem seu valor alterado, pois ela apenas é utilizada para a passagem por valor.

Mas em situações em que a variável necessita ter seu valor modificado pelo código da função que está sendo chamada, é preciso definir na função a passagem por referência que além de passar o valor, a própria variável também é passada para a função.

Para definir parâmetro com passagem por referência deve-se acrescentar "&" antes do nome do parâmetro:

funcao tipo nome_função (tipo **¶metro1**, tipo **¶metroN**)

No exemplo da Figura 7, a variável **cont** tem seu valor modificado pela função **contar** cada vez que a variável é passada para a função, já que na declaração da função seu parâmetro está definido por referência. Como a função **contar** soma 1 no parâmetro **&n**, que possui também a variável **cont**, esse novo valor é passado para a variável **cont** que continua sendo utilizada na função **inicio** com seu conteúdo modificado pela instrução dentro da função.

Figura 7: Exemplo de função com passagem por referência

```
programa
{
    funcao inicio()
    {
        // declaração e inicialização da variável cont
        inteiro cont = 0
        escreva("\n",cont)

        // chama a função pela primeira vez
        contar(cont)
        escreva("\n",cont)

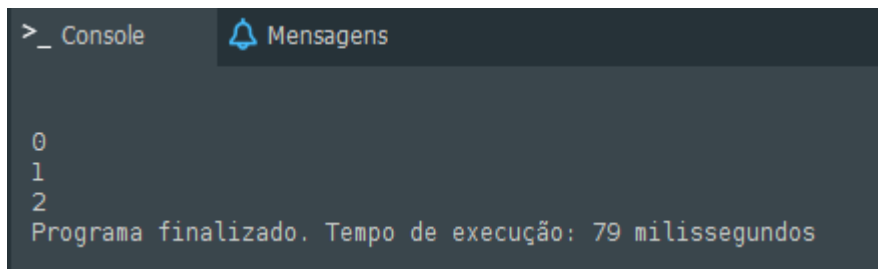
        // chama a função pela segunda vez
        contar(cont)
        escreva("\n",cont)
    }

    funcao contar(inteiro &n)
    {
        n++
    }
}
```

Fonte: Elaborado pela autora

A Figura 8 apresenta o resultado do comando **escreva()** do exemplo da Figura 7, demonstrando a modificação do conteúdo da variável **cont** após ser passada por referência para a função **contar(cont)**.

Figura 8: Console após execução do exemplo de passagem por referência



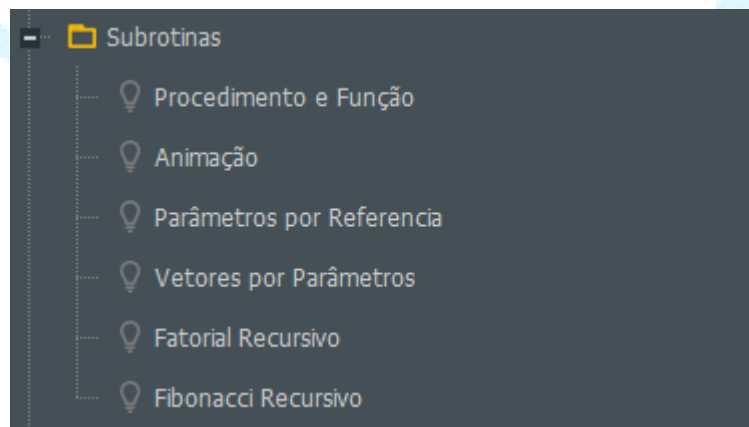
```
>_ Console Mensagens

0
1
2
Programa finalizado. Tempo de execução: 79 milissegundos
```

Fonte: portugol-studio.jar, 2022

No Portugol Studio, há exemplos de programas com funções, encontrados em Exemplos -> Subrotinas (Figura 9).

Figura 9: Algoritmos com vetores e matrizes do Portugol Studio



Fonte: portugol-studio.jar, 2022

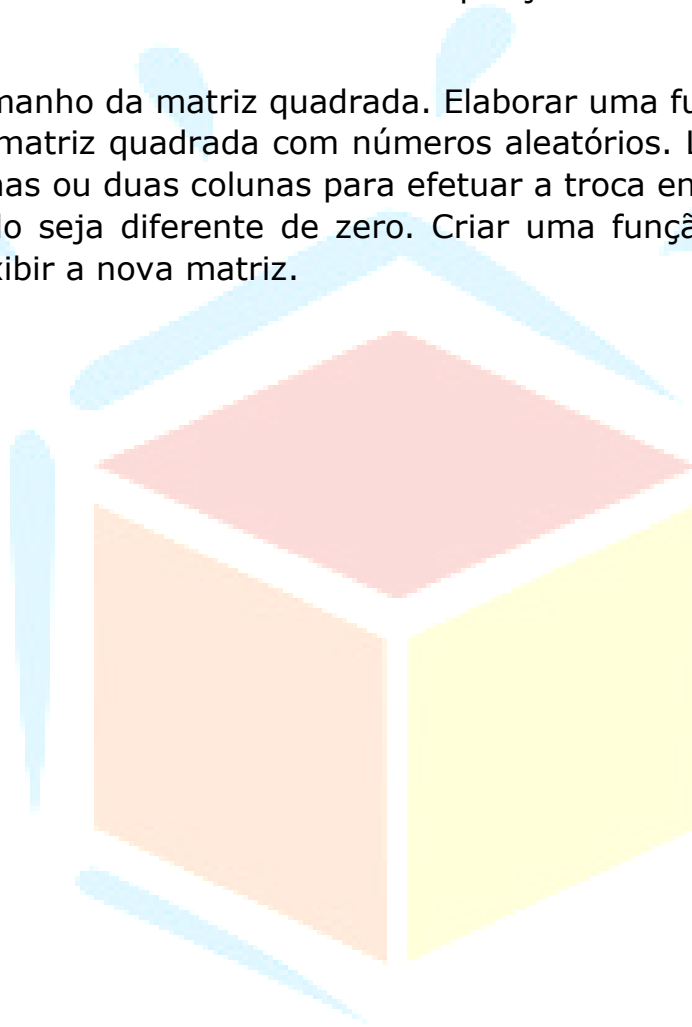
VOCÊ SABIA?

As funções que não possuem retorno podem ser chamadas de procedimento. Então, uma função é a subrotina que possui retorno, enquanto que o procedimento é a subrotina que não retorna nenhum valor.

PARA PRATICAR

1. Ler dois números inteiros e efetuar os seguintes cálculos: adição, subtração, multiplicação e divisão. Criar função para cada um dos cálculos. Exibir o resultado de todos os cálculos.

2. Ler um número inteiro. Exibir a soma dos números ímpares até o número lido. Criar uma função que retorne verdadeiro para números ímpares.
3. Ler o sexo e altura. Calcular e exibir o peso ideal. Criar uma função para o cálculo quando o sexo for masculino utilizando a fórmula $72,7 \times \text{altura} - 58$ e outra função para o sexo feminino utilizando a fórmula $62,1 \times \text{altura} - 44,7$.
4. Ler a base e o expoente. Criar uma função para efetuar o cálculo para potência utilizando estrutura de repetição. Exibir o resultado do cálculo.
5. Ler o tamanho da matriz quadrada. Elaborar uma função para criar e exibir a matriz quadrada com números aleatórios. Ler o número das duas linhas ou duas colunas para efetuar a troca enquanto o número informado seja diferente de zero. Criar uma função para efetuar a troca. Exibir a nova matriz.



REFERÊNCIAS

ALVES, Gustavo Furtado de Oliveira. **O que são Funções e Procedimentos?** Disponível em: <<https://dicasdeprogramacao.com.br/o-que-sao-funcoes-e-procedimentos/>> Acesso em 08 abr. 2022.

BRANDÃO, Leônidas de Oliveira. **Introdução ao uso de funções.** Disponível em: <https://www.ime.usp.br/~leo/mac2166/2017-1/introducao_funcoes.html> Acesso em 08 abr. 2022.

GASPAR, Wagner. **Como criar uma função que recebe parâmetros em Portugal?** Wagner Gatar, 2021. Disponível em: <<https://wagnergaspar.com/como-criar-uma-funcao-que-recebe-parametros-em-portugol/>>. Acesso em 09 abr. 2022.

GASPAR, Wagner. **Passagem de parâmetro por VALOR e por REFERÊNCIA em Portugal** Wagner Gatar, 2021. Disponível em: <<https://wagnergaspar.com/passagem-de-parametros-por-valor-e-por-referencia-em-portugol/>>. Acesso em 10 abr. 2022.

PINHO, Márcio Sarroglia. **Subalgoritmos (Funções).** GRU. Disponível em: <<https://www.inf.pucrs.br/~pinho/LaproI/Funcoes/AulaDeFuncoes.htm>>. Acesso em 09 abr. 2022.