Funções

- Definição : Sequência de instruções executadas somente quando chamadas por um programa em execução
  - Devem executar **uma tarefa** específica
  - Um programa **pode conter diversas funções**, além da função principal **início()** , que é **obrigatória**
  - As funções executam **somente** quando chamadas à partir da função inicio()
  - Após a execução, o fluxo retorna ao ponto **imediatamente após** o da chamada da função
  - Uma função pode (ou não) **retornar um valor** ao bloco que a chamou
  - Uma função pode (ou não) **necessitar de um ou mais argumentos** ao ser chamada

Vamos olhar uma etapa de cada vez …

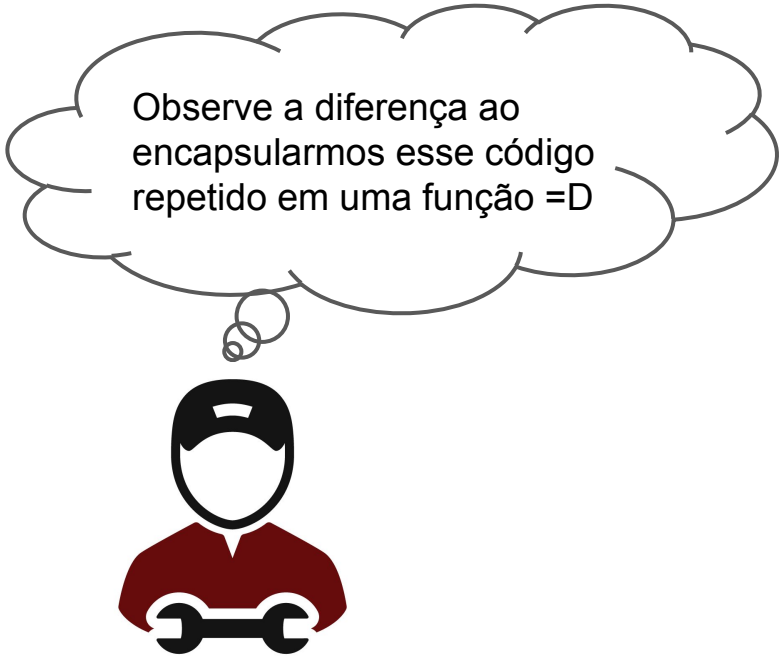
Mais alguns exemplos - Repetição de código

```
programa {
        funcao inicio(){
                inteiro i
                para(i=0; i<20; i++)
                        escreva(“*”)
                escreva(“\n”)
                escreva(“Numeros entre 1 e 5\n”)
                para(i=0; i<10; i++)
                        escreva(“*”)
                escreva(“\n”)
                para(i=1; i<=5; i++)
                        escreva(i,“\n”)
                para(i=0;i<20;i++)
                        escreva(“*”)
                escreva(“\n”)
        }
}
```

Note o código repetido. Se tivermos que consertar, teremos que fazer o mesmo ajuste várias vezes

Mais alguns exemplos - Repetição de código

```
programa {
	funcao inicio(){
		inteiro i
		escreve_linha()
		escreva("Numeros entre 1 e 5\n")
		escreve_linha()
		para(i=1; i<=5; i++)
			escreva(i,"\n")
		escreve_linha()
	}
	funcao escreve_linha(){
		para(i=0; i<20; i++)
			escreva("*")
		escreva("\n")
	}
}
```



Mais alguns exemplos - Recursividade

- Podemos também fazer a função chamar ela mesma para resolvermos problemas chamados **recursivos**.

Mas o que é recursão?

Recursividade

- Recursão é a definição de algo a partir dele mesmo

Recursividade

- Em Matemática e Ciência da Computação, uma classe de métodos tem comportamento recursivo quando eles podem ser definidos por duas propriedades:
  - Um caso base simples (ou vários casos)
  - Um conjunto de regras que reduz todos os outros casos para o caso base

Exemplo : Fatorial de um número inteiro positivo!!

5 * (fatorial de 4)
5! = 5 * 4 * 3 * 2 * 1

Fatorial Recursivo

```
programa {
    funcao inteiro fatorial(inteiro n){
        se(n == 0){
            retorne 1
        } senao {
            retorne n * fatorial( n - 1 )
        }

    }
}
```

Execução : 4 fatorial

fatorial(4) -> 4 * 3* 2 *1

n = 4

retorne 4 * fatorial(3)

n = 3

retorne 3 * fatorial(2)

n = 2

retorne 2 * fatorial(1)

n = 1

retorne 1 * fatorial(0)

retorne 1

Funções de bibliotecas

- Nós vimos várias funções como **escreva(), leia(), limpa().**
- Estas funções são métodos padrões já disponíveis em qualquer programa do PortugolStudio. Além dessas funções, podemos adicionar outras funções através da importação de bibliotecas.

```
programa
{
	inclua biblioteca Matematica --> mat
	funcao inicio()
	{
		real numero = 4.0
		real raiz = mat.raiz(numero, 2.0) // Obtém a raíz quadrada do número
		escreva("A raíz quadrada de ", numero , " é: ", raiz, "\n")
	}
}
```

Exercícios - Funções

1) Crie um programa com uma função chamada dobro que receba um número inteiro como parâmetro e retorne o dobro desse número. O programa principal deve solicitar um número ao usuário, chamar a função e exibir o resultado.

2) Em seguida, desenvolva uma função chamada mediaNotas que receba três notas do tipo real e retorne a média dessas notas. No programa principal, peça as três notas ao usuário, chame a função e mostre se o aluno está aprovado (média maior ou igual a 6) ou reprovado.

3) Crie também uma função chamada ehPar que receba um número inteiro e retorne verdadeiro se ele for par, ou falso se for ímpar. O programa principal deve ler um número e exibir o resultado da verificação.

4) Depois, implemente uma função chamada maiorNumero que receba dois números inteiros e retorne o maior entre eles. O programa deve exibir o valor retornado por essa função.

5) Implemente ainda uma função chamada fatorial que receba um número inteiro positivo e retorne o seu fatorial. No programa principal, solicite esse número ao usuário, chame a função e mostre o resultado do cálculo.

6) Crie uma função chamada mostrarMenu que não recebe parâmetros e apenas exibe um menu com três opções: “1 - Iniciar”, “2 - Configurações” e “3 - Sair”. No programa principal, chame essa função e leia a opção escolhida pelo usuário.

7) Desenvolva uma função chamada somarVetor que receba um vetor contendo 5 números inteiros e retorne a soma de todos os elementos. No programa principal, leia os 5 números, chame a função e exiba o total da soma.

Exercícios - Funções recursivas

1)Contagem regressiva:
Crie uma função recursiva que imprima uma contagem regressiva a partir de um número n fornecido pelo usuário até o número zero. Ao final da contagem, a função deve exibir a mensagem “Fim!”.

2) Soma recursiva até N:
Implemente uma função recursiva que receba um número inteiro positivo n e retorne a soma de todos os números inteiros de 1 até n.

3) Fatorial:
Desenvolva uma função recursiva que calcule o fatorial de um número inteiro n, onde o fatorial de n (representado por n!) é definido como n * (n-1) * (n-2) * ... * 1.

4) Contar dígitos de um número:
Crie uma função recursiva que receba um número inteiro positivo e retorne a quantidade de dígitos desse número.

5) Potência recursiva:
Desenvolva uma função recursiva que receba dois números inteiros, base e expoente, e calcule o valor de base elevado à potência do expoente, ou seja, base ^ expoente.

6) MDC (Máximo Divisor Comum):
Implemente uma função recursiva utilizando o algoritmo de Euclides para calcular o Máximo Divisor Comum entre dois números inteiros positivos fornecidos pelo usuário.