UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

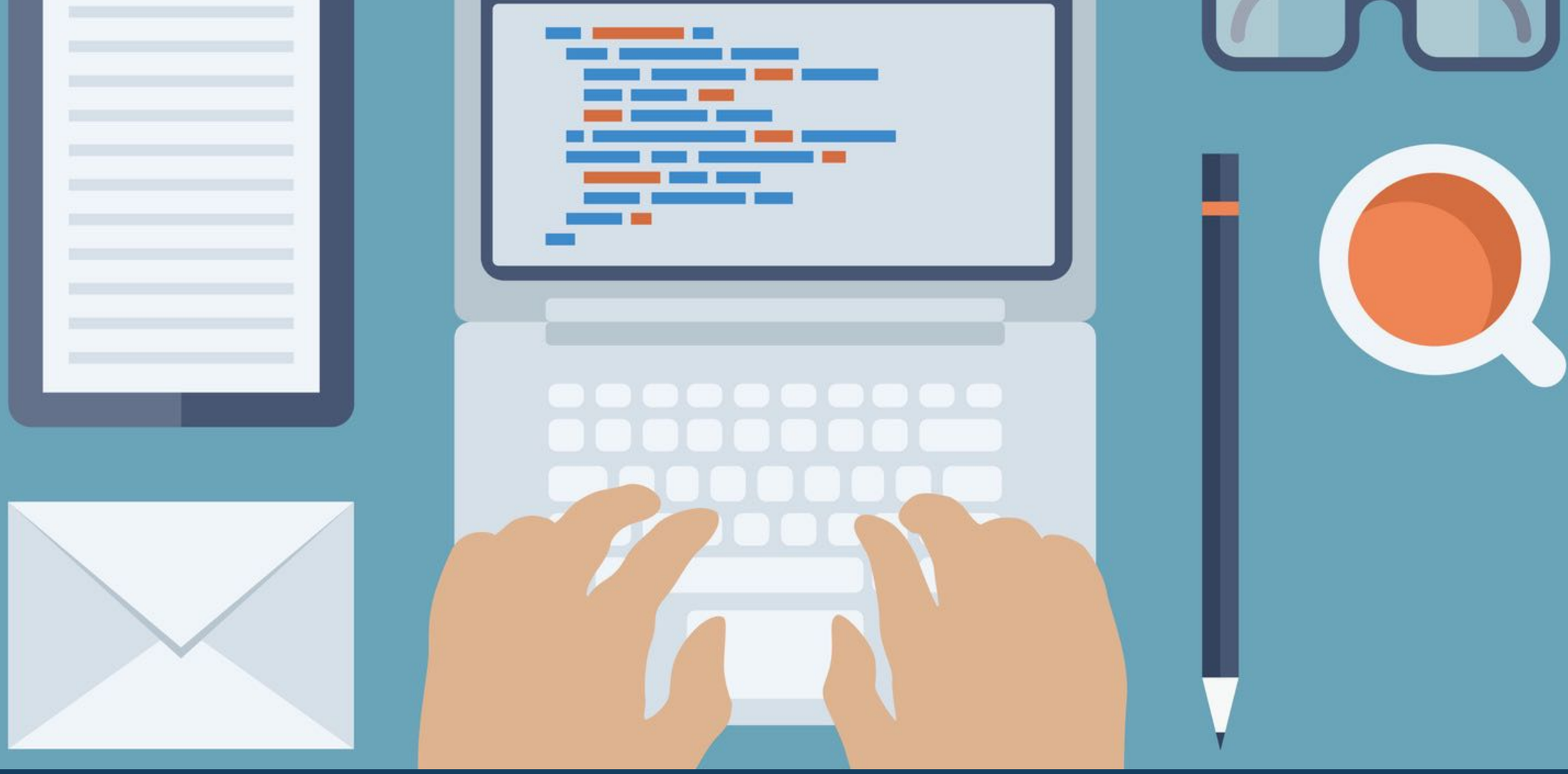
CK0211 - Fundamentos de Programação: Funções Recursivas e Arquivos

Emanuele Santos

Bibliografia: Ascencio, Cap. 11

Objetivos

- Aprender a implementar e utilizar funções recursivas
- Aprender a ler e a escrever para arquivos



FUNÇÕES RECURSIVAS

Definição

- Uma função pode chamar a si mesma
- Quando isso ocorre, dizemos que a função é recursiva.
- Vejamos o problema do fatorial:
  - Podemos definir o fatorial de um número não negativo n como sendo esse número multiplicado pelo fatorial de seu antecessor.
  - Definição recursiva:

$$\text{fatorial}(n) = \begin{cases} 1 & \text{, se } 0 <= n <= 1 \\ n \times \text{fatorial}(n-1) & \text{, caso contrário} \end{cases}$$

Fatorial em Python

```
def fatorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fatorial(n-1)

n = int(input("Digite o número para calcular o fatorial: "))
print(fatorial(n))
```

A sequência de Fibonacci

- A sequência de Fibonacci é outro problema clássico no qual podemos aplicar funções recursivas.
- A sequência começa com dois números 0 e 1
- Os números seguintes são a soma dos dois números anteriores.

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

A sequência de Fibonacci

- A função para calcular o enésimo termo da sequência de Fibonacci:

$$\text{fibonacci}(n) = \begin{cases} n & \text{, se } n <= 1 \\ \text{fibonacci}(n-1) + \text{fibonacci}(n-2) & \text{, caso contrário} \end{cases}$$

Fibonacci em Python

```
def fibonacci(n):
    if n <=1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

n = int(input("Digite um número maior ou igual a 0: "))
print(fibonacci(n))
```

Maior Divisor Comum (M.D.C)

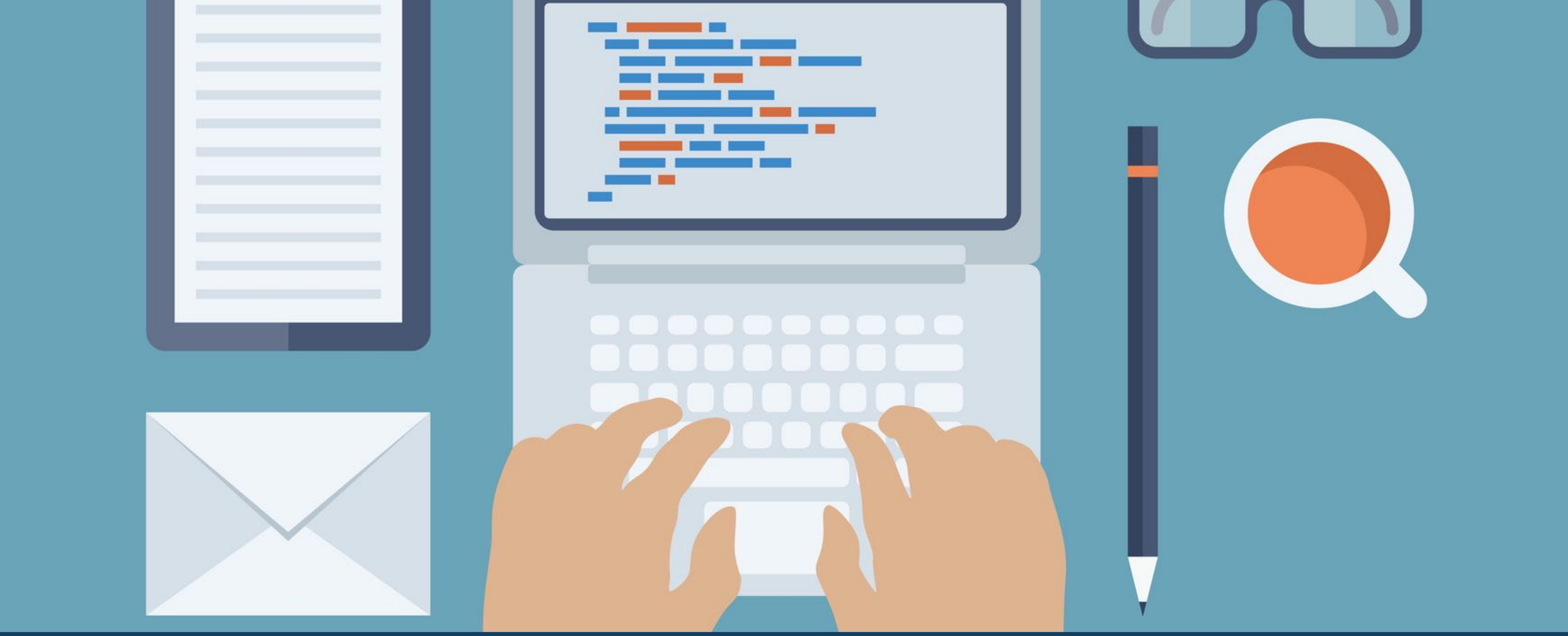
- O M.D.C. de dois números a e b, onde a > b, pode ser definido como:

$$MDC(a, b) = \begin{cases} a & , \text{se } b = 0 \\ MDC(b, a \% b), & \text{caso contrário} \end{cases}$$

MDC em Python

```
def mdc(a, b):
    if b == 0:
        return a
    else:
        return mdc(b, a%b)
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
if a < b:
    aux = b
    b = a
    a = aux

print("O M.D.C(%d, %d) = %d." % (a, b, mdc(a,b)))
```



ARQUIVOS

Motivação

- Precisamos de uma forma de armazenar dados permanentemente
- Além disso, precisamos de uma forma simples de entrada e saída de dados para programas
- A maneira mais simples de se fazer isso é através de **arquivos**

Arquivos

- Um arquivo é uma área em um disco, gerenciada pelo sistema operacional, onde podemos ler e gravar informações

- Do ponto de vista do programa, só precisamos saber do seu nome e como os dados serão lidos e/ou gravados (texto ou binário)

Abrindo arquivos

- Para acessar um arquivo, antes de mais nada, precisamos abri-lo
- Durante a abertura, dizemos o **nome do arquivo** (inclusive com o nome do diretório onde ele está) e o tipo de operação que queremos realizar: leitura e/ou escrita
- Em Python abrimos arquivos com a função **open** e ela retorna um objeto do tipo **file**

```
arquivo = open( nome_do_arquivo, modo)
```

Modos de abertura

Modo	Operações
r	leitura
w	escrita, apaga o conteúdo se já existir
a	escrita, mas preserva o conteúdo se já existir
b	modo binário
+	atualização (leitura e escrita)

Alguns modos podem ser combinados: “r+”, “w+”, “a+”, “r+b”, “w+b”

https://docs.python.org/3/library/functions.html#open

Leitura e Escrita

- Depois de abrir um arquivo e obter uma variável do tipo file, usamos as seguintes funções:
  - **arquivo.write(conteudo)** para escrever o conteúdo no arquivo
  - conteudo = **arquivo.read()** lê todo o conteúdo de arquivo de uma vez como uma string
  - linhas = **arquivo.readlines()** lê todo o conteúdo de arquivo de uma vez, separando cada linha de texto em uma posição da lista linhas

Leitura de arquivos

- Os arquivos só podem ser lidos uma vez

```
>>> arquivo = open("exemplo.txt",'r')
# Ler todo o arquivo de uma só vez
>>> arquivo.read()
'Isso deve ser o arquivo inteiro.\n'
>>> arquivo.read()
''
```

- Para ler um arquivo outra vez, feche-o e abra-o novamente antes de lê-lo

Leitura de arquivos

- Utiliza-se readlines() para retornar todas as linhas do arquivo em uma lista

```
>>> arquivo2 = open('exemplo2.txt')
>>> arquivo2.readlines()
['Essa e a primeira linha do arquivo\n','Essa e a segunda
linha do arquivo\n']
```

Leitura de arquivos

- Outra maneira de ler as linhas de um arquivo é através do comando for

```
>>> arquivo2 = open("exemplo2.txt",'r')
>>> for linha in arquivo2:
...     print(linha)
...
Essa e a primeira linha do arquivo

Essa e a segunda linha do arquivo
```

Fechamento

- Ao finalizar o processamento do arquivo, fechamos o arquivo:
  - **arquivo.close()**
- O fechamento é muito importante, pois ao fechar um arquivo garantimos que outros processos podem acessar o mesmo arquivo sem corrompê-lo.

Trabalhando com arquivos

- 1. Abertura
- 2. Leitura/Escrita
- 3. Fechamento

Abrindo, escrevendo e fechando um arquivo

```
arquivo = open("numeros.txt", "w")
for linha in range(1,101):
    arquivo.write("%d\n" % linha)
arquivo.close()
print("Fim.")
```

O programa acima irá abrir um arquivo chamado numeros.txt no diretório atual e vai escrever um número para cada linha do arquivo, de 1 a 100

Se numeros.txt não existir, será criado. Se já existir, seu conteúdo será apagado antes dos números das linhas serem escritos

Abrindo, lendo e fechando um arquivo

```
arquivo = open("numeros.txt", "r")
for linha in arquivo.readlines():
    print(linha)
arquivo.close()
print("Fim.")
```

O programa acima irá abrir um arquivo chamado numeros.txt no diretório atual e vai ler o conteúdo de cada linha e mostrar na tela.

Exercício 1

- Escreva um programa que gera dois arquivos com 500 linhas cada. No primeiro arquivo o programa escreve apenas os números pares de 1 a 1000 e no outro os ímpares.

Exercício 2

- Ler um arquivo de notas (notas.txt) e mostrar a média (nesse arquivo cada nota é inteira, varia de 0 a 100 e ocupa uma linha)

notas.txt

```
60
100
76
50
95
75
63
...
```

Exercício 3

- Modifique o programa anterior para mostrar, além da média, a menor e a maior nota.

notas.txt

```
60
100
76
50
95
75
63
...
```