UNIVERSIDADE FEDERAL DO CEARÁ

DEPARTAMENTO DE COMPUTAÇÃO

CIÊNCIA DA COMPUTAÇÃO

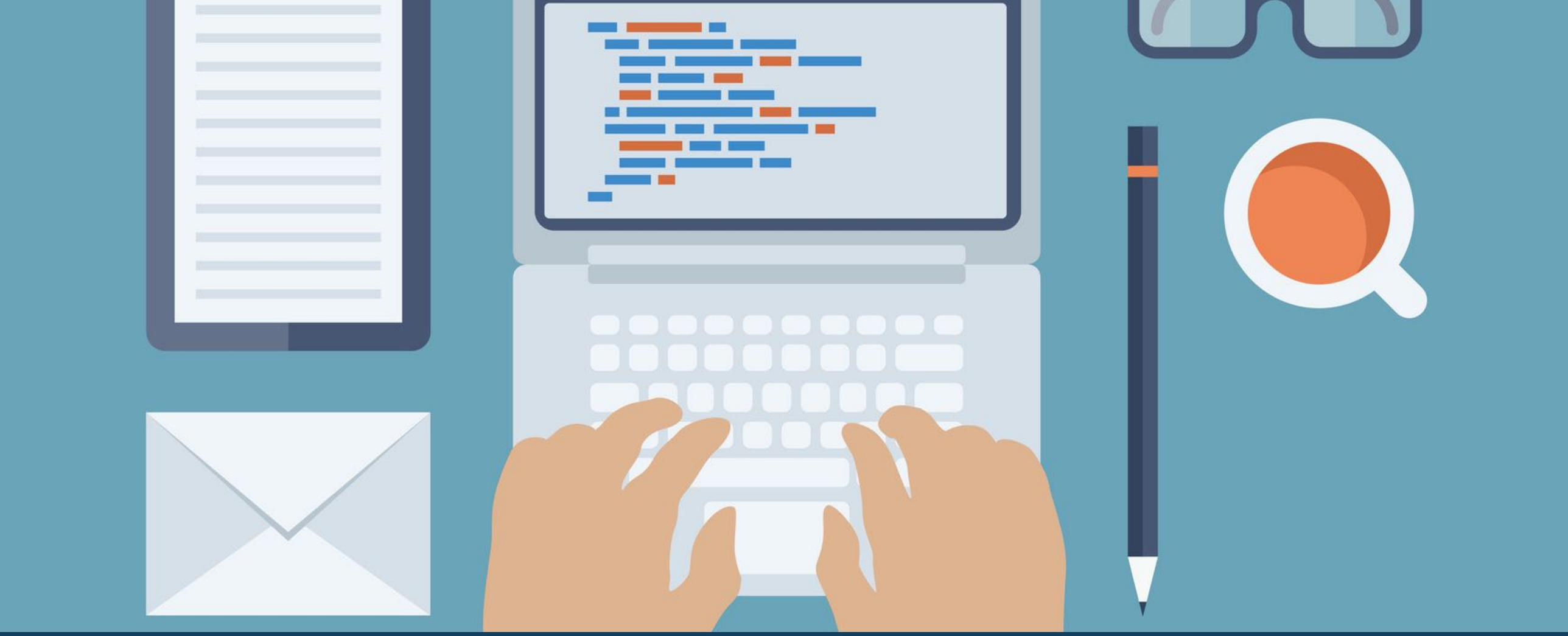
CK0211 - Fundamentos de Programação: Funções ou Sub-rotinas

Emanuele Santos

Bibliografia: Ascencio, Cap. 8

Objetivos

- Aprender a decompor algoritmos em problemas menores e resolvê-los por partes



SUB-ROTINAS EM ALGORITMOS

Introdução

- Um problema complexo pode ser simplificado quando dividido em vários problemas menores
- Decomposição
  - Redução de complexidade
  - Permite focalizar a atenção em um problema pequeno de cada vez
  - Produz melhor compreensão do todo

Decomposição

- Analogia com o corpo humano
  - Decomposição em sistemas (digestivo, respiratório, nervoso, cardiovascular, etc.)
  - Sistemas por sua vez são decompostos em órgãos
- Ideia: decompor algoritmos em módulos ou sub-rotinas ou funções ou subprogramas responsáveis por pequenas tarefas

Sub-rotinas ou funções ou subprogramas

- As sub-rotinas são blocos de instruções que realizam tarefas específicas
- O código de uma sub-rotina é carregado uma vez e pode ser executado quantas vezes forem necessárias
- Programação Modularizada
  - Subdividindo o problema em pequenas tarefas, os **programas** tendem a ficar **menores** e **mais organizados**

Sub-rotinas ou funções ou subprogramas

- Os programas em geral são executados linearmente, uma linha após a outra, até o fim
- As funções permitem a realização de desvios na execução dos programas
  - Desvios são efetuados quando uma função é chamada pelo programa principal

Sub-rotinas: Exemplo 1

- O Algoritmo a seguir tem como objetivo receber o valor atual do salário de um funcionário e calcular o novo salário

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA “Novo salário é ”, novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

Programa principal

Sub-rotina

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA "Novo salário é ", novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

parâmetro

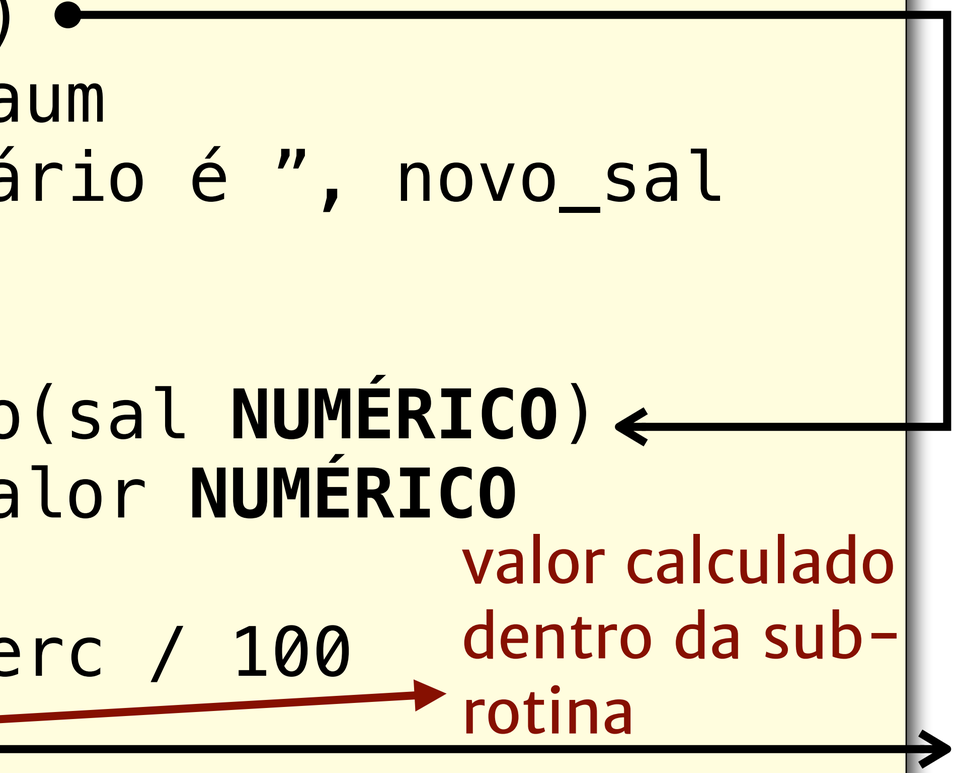
O programa principal é executado linearmente até a linha 4.

Chamada à sub-rotina **calculo** (programa principal fica temporariamente suspenso)

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA "Novo salário é ", novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```



O programa principal é executado linearmente até a linha 4.

Com a chamada da sub-rotina, ocorre um desvio da execução das instruções para a linha 8.

A execução só volta ao programa principal quando o comando RETORNE for executado.

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA “Novo salário é ”, novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

A execução do programa principal é retomada exatamente no ponto em que foi interrompida e dessa maneira, o valor devolvido pela sub-rotina é atribuído à variável aum (linha 4).

Sub-rotinas

- No exemplo anterior, a sub-rotina calculo recebe um parâmetro (o valor atual do salário) e retorna um valor (aumento que será dado ao salário) para quem a chamou
- Sub-rotinas podem não receber parâmetros
- Sub-rotinas podem não retornar valor algum
  - Nesse caso elas não devem aparecer do lado direito de uma expressão de atribuição

Variáveis locais e globais

- Quando há declaração de variáveis dentro de uma sub-rotina, essas variáveis são chamadas de variáveis **locais**
  - São visíveis apenas dentro da sub-rotina
  - Quando a execução da sub-rotina chega ao fim, essas variáveis são destruídas e seus conteúdos são perdidos
- Variáveis declaradas fora de qualquer sub-rotina são chamadas **globais**
  - São visíveis em qualquer ponto do programa, inclusive dentro das sub-rotinas

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA “Novo salário é ”, novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

Variáveis globais: visíveis em todo o programa

Variáveis locais: visíveis apenas dentro da sub-rotina calculo

Variáveis locais e globais

- Sempre que possível, **evite o uso de variáveis globais**: elas dificultam a manutenção e a busca por erros nos programas

Parâmetros em sub-rotinas

- Na chamada da sub-rotina calculo, o conteúdo da variável sal (declarada no programa principal) está sendo passada como um valor para o parâmetro sal definido no cabeçalho da sub-rotina (linha 8)
  - Note que tanto o parâmetro quanto a variável poderiam ter nomes diferentes!

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA “Novo salário é ”, novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

Variável sal sendo passada como parâmetro

Nome do parâmetro sal

Normalmente, os parâmetros se comportam como variáveis locais

Tipos de passagem de parâmetros

- **Passagem de parâmetros por valor**
  - Variável do cabeçalho da sub-rotina se comporta como uma variável local e qualquer alteração no parâmetro só é visível dentro da sub-rotina
  - Depois que a sub-rotina é finalizada, a variável que foi passada como parâmetro por valor contem o valor que tinha no momento da chamada da função

Exemplo 1

```
1.  ALGORITMO
2.  DECLARE sal, aum, novo_sal NUMÉRICO
3.  LEIA sal
4.  aum ← calculo(sal)
5.  novo_sal ← sal + aum
6.  ESCREVA “Novo salário é ”, novo_sal
7.  FIM_ALGORITMO.

8.  SUB-ROTINA calculo(sal NUMÉRICO)
9.    DECLARE perc, valor NUMÉRICO
10.   LEIA perc
11.   valor ← sal * perc / 100
12.   RETORNE valor
13. FIM_SUB_ROTINA calculo
```

Variável sal sendo passada como parâmetro por valor

Mesmo que sal fosse alterada dentro da sub-rotina, ela não alteraria o valor da variável sal global.

Tipos de passagem de parâmetros

- **Passagem de parâmetros por referência**
  - Variável do cabeçalho da sub-rotina se comporta como uma variável global (as alterações são permanentes)
- O que vai diferenciar o tipo de passagem (por valor ou por referência) depende da sintaxe de cada linguagem de programação
- Em algumas linguagens a sintaxe é explícita (usam-se símbolos para diferenciar o tipo de passagem de parâmetros)

pass by reference

cup =

fillCup( )

pass by value

cup =

fillCup( )

www.penjee.com

http://devblog.drall.com.br/wp-content/uploads/2016/06/pass-by-reference-vs-pass-by-value-animation.gif

Passagem por valor e por referência em C

```
void incrementa1_valor(int num) {
  num = num + 1;
  printf(“%d”,num);
}
```

```
void incrementa1_ref(int *num) {
  *num = *num + 1;
  printf(“%d”,*num);
}
```

```
int a = 5;
incrementa1_valor(a);
printf(“%d”,a);
incrementa1_ref(&a);
printf(“%d”,a);
```

Exemplo 2 (parte 1)

```
1.  ALGORITMO
2.  DECLARE X, Y NUMÉRICO
3.  X ← 1
4.  Y ← 2
5.  ESCREVA “VALORES INICIAIS”
6.  ESCREVA X, Y
7.  S1
8.  ESCREVA “VALORES DEPOIS DE S1”
9.  ESCREVA X, Y
10. S2(X,Y)
11. ESCREVA “VALORES DEPOIS DE S2”
12. ESCREVA X, Y
13. S3(X,Y)
14. ESCREVA “VALORES DEPOIS DE S3”
15. ESCREVA X, Y
16. FIM_ALGORITMO.
    …
```

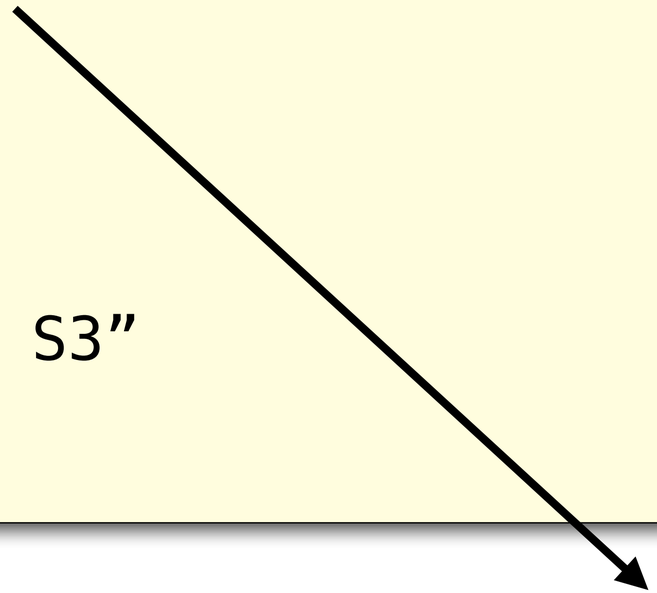
Exemplo 2 (parte 2)

```
17. SUB_ROTINA S1
18.   DECLARE X, Y, Z NUMÉRICO
19.   X ← 8
20.   Y ← 10
21.   Z ← 5
22.   ESCREVA “VALORES DENTRO DE S1”
23.   ESCREVA X, Y, Z
24. FIM_SUB_ROTINA S1

25. SUB_ROTINA S2(X, Y NUMÉRICO)
26.   DECLARE Z NUMÉRICO
27.   X ← X + 2
28.   Y ← Y * 2
29.   Z ← X + Y
30.   ESCREVA “VALORES DENTRO DE S2”
31.   ESCREVA X, Y, Z
32. FIM_SUB_ROTINA S2
```

Exemplo 2 (parte 3)

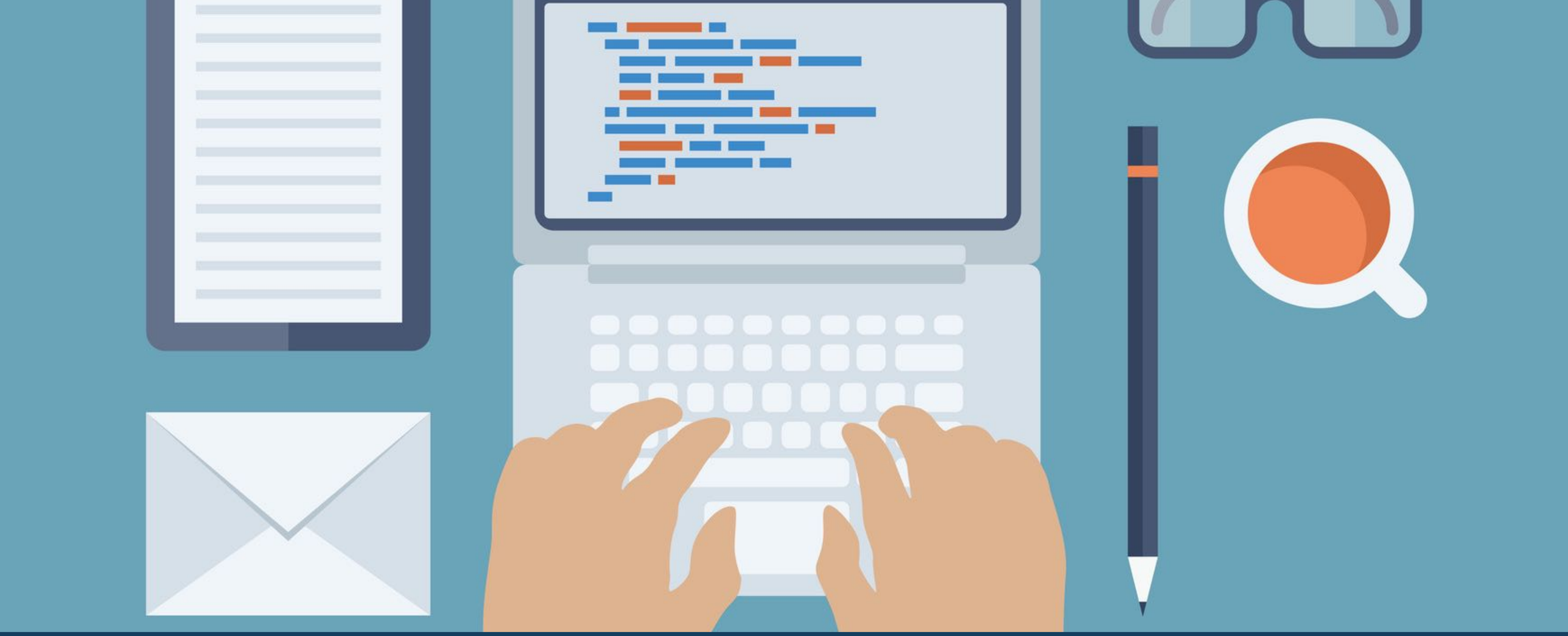
```
33. SUB_ROTINA S3(X, Y NUMÉRICO)
34.   DECLARE A NUMÉRICO
35.   A ← X + Y
36.   X ← X – 1
37.   Y ← Y – 2
38.   ESCREVA “VALORES DENTRO DE S3”
39.   ESCREVA X, Y, A
40. FIM_SUB_ROTINA S3
```



Usaremos sublinhado para denotar passagem por referência em pseudo-código

Exemplo 2 (saída do algoritmo)

```
VALORES INICIAIS
1 2
VALORES DENTRO DE S1
8 10 5
VALORES DEPOIS DE S1
1 2
VALORES DENTRO DE S2
3 4 7
VALORES DEPOIS DE S2
1 2
VALORES DENTRO DE S3
0 0 3
VALORES DEPOIS DE S3
0 0
```



SUB-ROTINAS EM PYTHON

Funções em Python

- Nós sabemos como usar algumas funções em Python:
  - print, input, float, int, range, …
- Veremos a seguir como definir uma nova função

Definição de funções em Python

```
def nome(parametro1, parametro2, …, parametroN):
    comando1
    …
    comandoN
    return valor # se a função retornar algum valor
```

- onde:
  - **nome** é o nome da função (com as mesmas restrições de um nome de variável)
  - parametro1, parametro2, …, parametro3 é uma lista de parâmetros
  - comando1, …, comandoN é o corpo da função, uma lista de comandos
  - o comando return é opcional

Exemplo 1

```
1. def calculo(sal):
2.     perc = float(input("Entre com o percentual de aumento: "))
3.     valor = sal * perc / 100
4.     return valor

5. sal = float(input("Entre com o salário: "))
6. aum = calculo(sal)
7. novo_sal = sal + aum
8. print("Novo salario é: R$%.2f" % novo_sal)
```

Exemplo 2

```
1. def eh_par(num):
2.     resultado = False
3.     if num % 2 == 0:
4.         resultado = True
       return resultado
5.
6. print(eh_par(2))
7. print(eh_par(3))
8. print(eh_par(8))
```

Tipos de passagem de parâmetros

- Python não diferencia o tipo de passagem de parâmetros em funções através da sintaxe
- A princípio, a passagem de parâmetros em Python é feita sempre por referência
- A passagem só será feita por valor quando o tipo do parâmetro for um tipo simples ou imutável (int, float, bool, str)
- Os demais tipos, tais como listas (vetores e matrizes) são passados sempre por referência

Exemplo 3

```
1. def incrementa1(num):
2.     num = num + 1
3.     print(“dentro de incrementa1:”,num)

4. a = 5
5. incrementa1(a)
6. print(a)
```

Exemplo 4

```
1. def incrementa1(num):
2.     num = num + 1
3.     print(“dentro de incrementa1:”,num)
4.     return num

5. a = 5
6. a = incrementa1(a)
7. print(a)
```

Exemplo 5

- Faça uma função (cria_matriz) que cria uma matriz m por n, preenchida com zeros e uma função (print_matriz) que mostra uma matriz de qualquer dimensão
- Crie um programa que utiliza a função cria_matriz para criar uma matriz 4 por 3 e usa a função print_matriz para mostrar os seus valores na tela

Código em Python:


```
#coding: utf-8
def cria_matriz(m, n):
    mat = []
    for i in range(m):
        linha = [0] * n
        mat.append(linha)
    return mat

def print_matriz(mat):
    for linha in mat:
        for elemento in linha:
            print("%8.2f" % elemento, end=" ")
        print("")

M = cria_matriz(4,3)
print_matriz(M)
```