FIAP ON

LÓGICA DE PROGRAMAÇÃO

FUNÇÕES, PROCEDIMENTOS, VETORES E MATRIZES

5

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

FUNÇÕES, PROCEDIMENTOS, VETORES E MATRIZES

1 INTRODUÇÃO

Até o capítulo 4, aprendemos os oito comandos de lógica de programação em Pseudocódigo, Python e Java. Essas instruções são conhecidas como comandos fundamentais porque cada um tem um objetivo e atende a uma necessidade específica. Para tudo que aprendermos a partir daqui, usaremos esses oito comandos de forma coadjuvante, mas não deixaremos de usá-los.

Quanto aos comandos de lógica de programação, pudemos ver a similaridade e as diferenças em cada linguagem, inclusive as suas sintaxes. Quando aprendemos outras linguagens de programação, veremos nelas esses oito comandos com sintaxes talvez diferentes, mas o que importa é a essência dos comandos.

Fazendo uma analogia, seria como se cada comando fosse uma ferramenta de obras. Dado um problema, temos que saber quais ferramentas e ordens que devemos utilizar, por exemplo, se precisamos montar um móvel, precisaremos de uma chave de fenda, um martelo, uma serra, uma chave Philips, lixa... De acordo com a fase da montagem, precisamos de determinadas ferramentas. Assim funciona a programação, dependendo do problema (algoritmo) que formos resolver, utilizamos as ferramentas aprendidas.

Antes de prosseguirmos, vamos sintetizar os oito comandos vistos até então:

Tabela 1 – Resumo dos oito comandos de programação no Pseudocódigo, Python e Java

Instrução	Pseudocódigo	Python	Java
Saída de dados	Escreva	print()	System.out.print();
Entrada de dados	Leia	input()	<var> = <obj>.next();
Processamento	Cálculos	a = b + c	a = b + c;
Se simples	Se <condição> então ...Fim_se	if <condição>: ...	if() ...;
Se composto	Se <condição> então ...Senão ...Fim_se	if <condição>: ... else: ...	if(<condição>) ...;else ...;
Se encadeado	Se <condição> então ...Senão se (condição) ...Senão ...Fim_se	if <condição>: ... elif <condição>: ... else: ...	if(<condição>) ...;else if(condição) ...;else ...
Escolha	Escolha Caso ... Senão ... Fim_escolha	Não existe	switch(){ case: ... default: ...}
Laço pré-condicional	Enquanto <cond> faça ... Fim_Enquanto	while <condição>: ...	while(<condição>) ...;
Laço pós-condicional	Faça ...Enquanto <condição>	Não existe	do{ ... }while(<condição>);
Laço contador	Para ... faça ... Fim_enquanto	For ...: ...	for(...) ...;

Fonte: Elaborada pelo autor (2022)

Vistos esses comandos, agora vamos subir um nível no aprendizado de lógica de programação. Veremos agora subalgoritmos (funções e procedimentos).

Vimos que algoritmo é a resolução estruturada de um problema por meio de passos em ordem lógica, logo, o prefixo “sub” pode ser definido como menor, contido ou subordinado. Sendo assim, subalgoritmo é a quebra da resolução do problema (algoritmo) em problemas menores (subalgoritmos).

Algoritmo é a representação estruturada do problema:

Figura 1 – Ilustração de um programa estruturado
Fonte: Elaborada pelo autor (2022)

Considere que o quadro vermelho representa o arquivo com o código-fonte escrito de forma estruturada, e as linhas amarelas as instruções do código-fonte.

Para criarmos subalgoritmos, devemos olhar para as instruções cuja finalidade é a mesma e separá-las, veja a Figura “Ilustração dos subalgoritmos no algoritmo” abaixo:

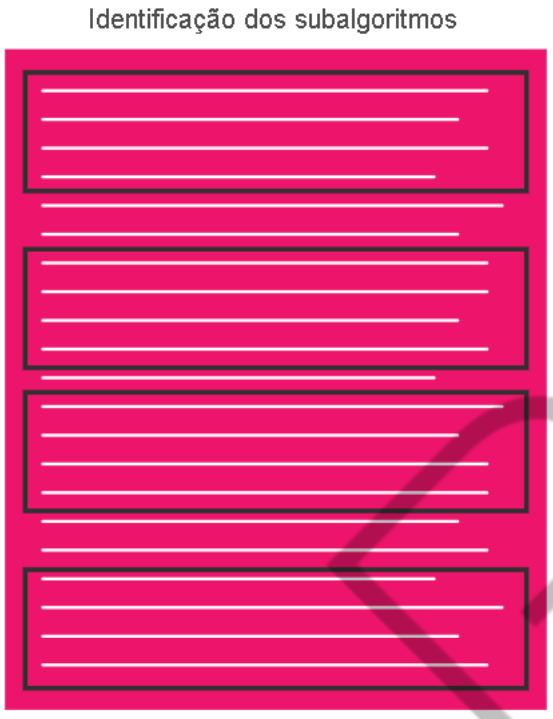

Figura 2 – Ilustração dos subalgoritmos no algoritmo
Fonte: Elaborada pelo autor (2022)

Uma vez identificados os subalgoritmos, devemos nomeá-los. Na Figura “Ilustração da nomeação dos subalgoritmos” abaixo, foram colocados nomes hipotéticos: A, B, C e D; mas a boa prática é a de colocarmos nomes que tenham a ver com o objetivo do bloco, tipo: calcular_media. Segue a ilustração:

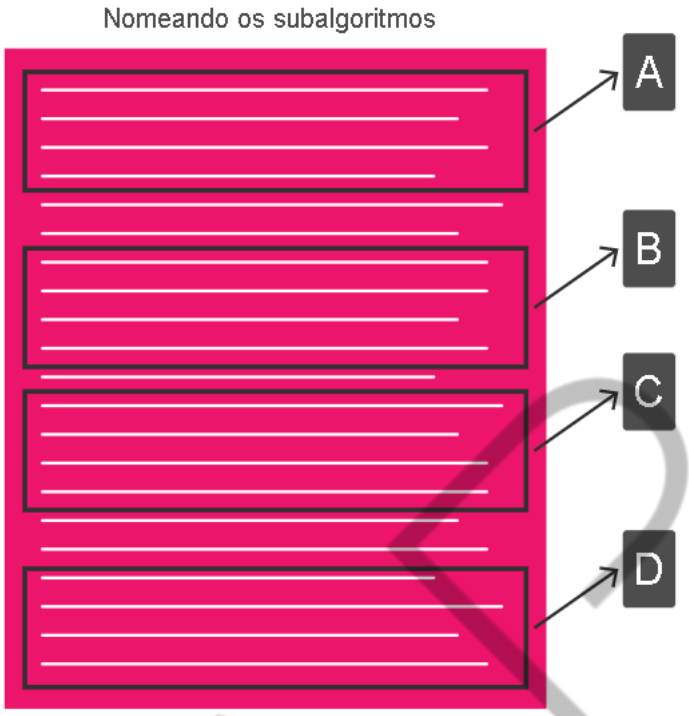

Figura 3 – Ilustração da nomeação dos subalgoritmos
Fonte: Elaborada pelo autor (2022)

Com os nomes dos subalgoritmos definidos, o ideal (mas não obrigação) é que os subalgoritmos sejam colocados em arquivos separados para uma melhor organização dos códigos. É assim que qualquer sistema profissional funciona:

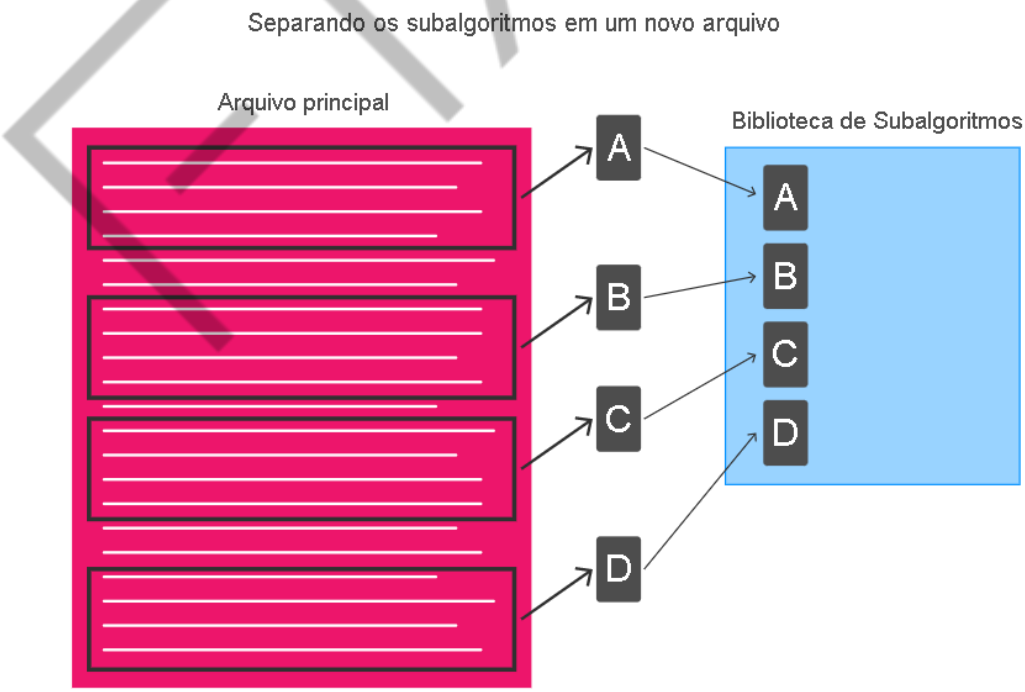

Figura 4 – Ilustração da separação dos subalgoritmos em uma biblioteca
Fonte: Elaborada pelo autor (2022)

O mecanismo de funcionamento desse conceito é o de, na chamada do subalgoritmo, o fluxo sair momentaneamente do programa principal e executar o subalgoritmo em um arquivo externo, depois retornar para a própria linha ou para a próxima (isso depende de ser função e procedimento). Veja a Figura “Ilustração do mecanismo de funcionamento do algoritmo com subalgoritmo”:

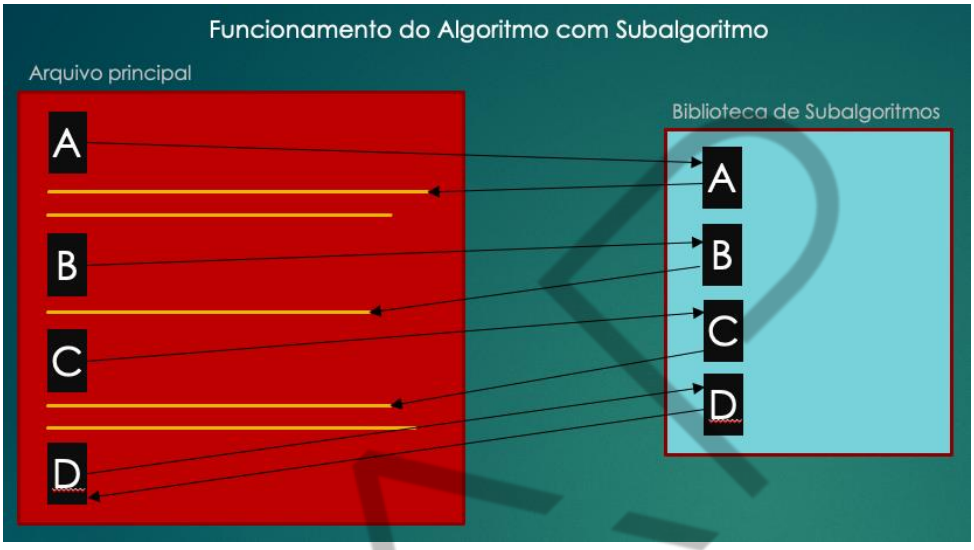

Figura 5 – Ilustração do mecanismo de funcionamento do algoritmo com subalgoritmo
Fonte: Elaborada pelo autor (2022)

No decorrer do capítulo, nós detalharemos as semelhanças e diferenças entre função e procedimento.

Outro assunto que abordaremos neste capítulo serão variáveis indexadas: vetor e matriz, também conhecidas como listas. Essas são classificações diferentes de variáveis de memória. A característica de variável de memória é a de armazenar UMA informação por variável enquanto as variáveis indexadas armazenam mais de um valor na mesma variável. Detalharemos esse conceito no final do capítulo.

2 PROCEDIMENTOS SEM PARÂMETROS

Dada a introdução sobre subalgoritmos, agora vamos detalhá-los.

Os subalgoritmos são representados por funções e procedimentos, com passagem de parâmetros ou não. Neste tópico, falaremos sobre procedimentos sem parâmetros.

Quando decidimos criar um subalgoritmo, consideramos, entre outras coisas, que ele será executado em outros momentos da aplicação ou em até outras aplicações.

Sem conhecer o conceito de procedimento, nós já o utilizávamos de forma empírica. Por exemplo, em Python, há um “comando” (que na verdade é um procedimento) que apaga a tela, veja:

```
import os
os.system('clear') or None
```

Código-fonte 1 – Exemplo de subalgoritmo predefinido no Python (1)
Fonte: Elaborado pelo autor (2022)

Para limpar a tela do console com Python, nós precisamos executar o comando `clear` utilizando o procedimento/método `system` da biblioteca/objeto padrão `os` (em outro momento falaremos sobre métodos e objetos). Em outras palavras, “alguém” criou um “comando” (que na verdade é procedimento) que apaga a tela. Basicamente quem o fez percorreu todas as linhas (padrão 25) com um for e todas as colunas (padrão 80) com outro for e exibiu “ “ (espaço) em cada movimentação do cursor.

Repare que, para o procedimento `os.system('clear')` funcionar, foi necessário importar a biblioteca `os`. Da mesma forma, quando criarmos os nossos subalgoritmos, podemos colocá-los em “bibliotecas”.

Ou, ainda, podemos usar como exemplo o print:

```
print("O print é um procedimento")
```

Código-fonte 2 – Exemplo de subalgoritmo predefinido no Python (2)
Fonte: Elaborado pelo autor (2022)

Quem criou o Python utilizou as instruções de Output (periféricos de saída) para que o programador consiga exibir alguma mensagem.

Vamos ver a visualização ilustrativa de um subalgoritmo que não passa parâmetro:

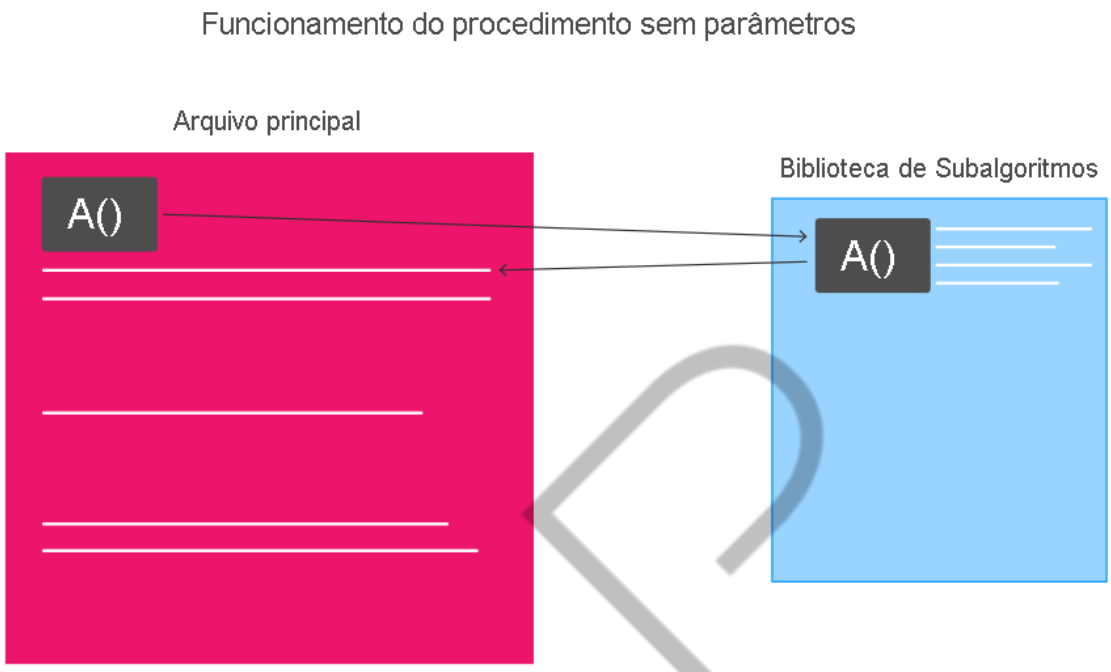

Figura 6 – Ilustração do mecanismo de funcionamento procedimento sem parâmetros
Fonte: Elaborada pelo autor (2022)

Repare: o “Arquivo Principal” está invocando o procedimento A() que está no arquivo “Biblioteca de Subalgoritmo” sem passar parâmetros (entre parênteses vazio). Executa-se o procedimento na Biblioteca e, ao voltar ao programa principal, é executada a linha seguinte.

Vamos criar o nosso primeiro procedimento? Antes, vamos a uma definição importante:

Procedimento é um tipo de subalgoritmo que executa as instruções que compõem um objetivo e que não retorna valor ao programa chamador.

Antes de criarmos o nosso procedimento, precisamos saber como são as sintaxes dele no Pseudocódigo, Python e Java:

Momento Pseudocódigo

```
Procedimento <nome do procedimento>()
Inicio
    <corpo do procedimento>
Fim
```

Momento Python

```
def <nome do Subalgoritmo>():
    corpo do procedimento
```

Momento Java

```
public static void <nome do procedimento>()
{
    <corpo do procedimento>;

}
```

Código-fonte 3 – Sintaxes de procedimento sem parâmetro em Pseudocódigo, Python e Java
Fonte: Elaborado pelo autor (2022)

Dadas as sintaxes, vamos agora criar um procedimento sem passagem de parâmetros:

Criar um procedimento com o nome “saudacao" que, ao ser invocado, exiba na tela: “Olá Usuário, você está logado”.

Depois de criar o procedimento, chamá-lo pelo programa principal.

Execução do exemplo:

```
Olá Usuário, você está logado
```

Quadro 1 – Execução do procedimento “saudacao” – sem parâmetros
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

```
Procedimento saudacao()
Início
    Escreva “Olá usuário, você está logado”
Fim

Programa testando_procedimento
Início
    saudacao();
Fim
```

Código-fonte 4 – Procedimento “saudacao” no pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

```
# Criação do procedimento
def saudacao():
  print("Olá Usuário, você está logado")
```

```
# Chamada do procedimento
saudacao()
```

Código-fonte 5 – Procedimento “saudacao” no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```
import java.util.Scanner;
public class First {
	// Criação do Procedimento
	public static void saudacao()
	{
		System.out.println("Olá Usuário, você está logado");
	}

	// Chamada do Procedimento
	public static void main(String[] args)
	{
		saudacao();
	}
}
```

Código-fonte 6 – Procedimento “saudacao” no Java
Fonte: Elaborado pelo autor (2022)

Apesar de, na teoria escrita, até então, sobre subalgoritmo, nós dizermos que ele “vai de um arquivo principal até outro biblioteca”, nestes exemplos iniciais criaremos os subalgoritmos dentro do arquivo principal.

3 PROCEDIMENTOS COM PARÂMETROS

Um procedimento com parâmetros tem basicamente a mesma definição do sem parâmetros. A diferença é que ele transposta valores do programa principal (ou programa chamador) via parâmetro até o subalgoritmo que está na biblioteca.

Parâmetros são informações passadas via chamada do subalgoritmo do programa principal para o subalgoritmo. Dentro do subalgoritmo, os parâmetros servem como variáveis.

Vamos ver a visualização ilustrativa de um subalgoritmo que passa parâmetro:

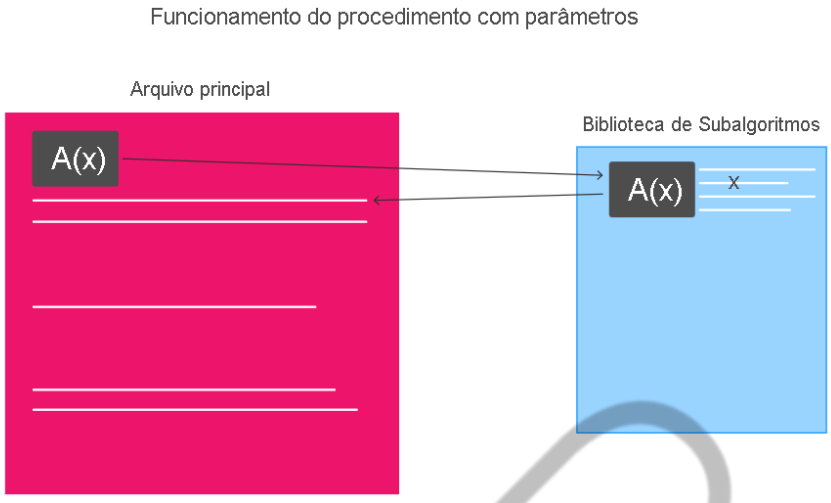

Figura 7 – Ilustração do mecanismo de funcionamento procedimento com parâmetros
Fonte: Elaborada pelo autor (2022)

O “Arquivo Principal” está invocando o procedimento A(x) que está no arquivo “Biblioteca de Subalgoritmo” passando parâmetro (o x entre parênteses). Executa-se o procedimento na Biblioteca, utilizando o parâmetro x em algum momento, e ao voltar ao programa principal é executada a linha seguinte.

A quantidade de parâmetros pode ser maior que um. Os tipos dos parâmetros podem ser diferentes.

As sintaxes dos procedimentos com parâmetros ou sem parâmetros são parecidas, apenas acrescentam-se os parâmetros:

Momento Pseudocódigo

```
Procedimento <nome do procedimento>([<parâmetros:tipos>])
Início
    <corpo do procedimento>
Fim
```

Momento Python

```
def <nome do Subalgoritmo>([<parâmetros>]):
    corpo do procedimento
```

Momento Java

```
public static void <nome do procedimento>(<tipo> <parâmetros>)
{
    <corpo do procedimento>;
}
```

Código-fonte 7 – Sintaxes de procedimento com parâmetro em Pseudocódigo, Python e Java
Fonte: Elaborado pelo autor (2022)

Em Java, o que caracteriza um procedimento é o retorno do tipo void (vazio) por obviamente um procedimento não retornar valor.

Baseando-se no exemplo anterior, vamos acrescentar parâmetros na mensagem de saudação. Para diferenciar do último exemplo, vamos chamar o procedimento de “saudacao2”:

Criar um procedimento com o nome “saudacao2" que, ao ser invocado, exiba na tela:

“Olá <nome do usuário>, <bom | dia | tarde | noite>! você está logado,”.

As informações <nome do usuário> e <hora> (para definir “Bom dia”, “Boa tarde” ou “Boa noite”) serão passadas por parâmetro pelo procedimento

Depois de criar o procedimento, chame-o pelo programa principal.

Execução do exemplo:

```
Olá Edson, boa tarde! Você está logado.
```

Quadro 2 – Execução do procedimento “saudacao2” – com parâmetros
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

```
Procedimento saudacao2(usuario: Texto, hora: inteiro)
Var
   msg: texto
Início
   Se hora < 12 então
       msg = “Bom dia!”
   senão se hora < 18 então
       msg = “Boa tarde!”
   senão
       msg = “Boa noite!”
   fim_se
   escreva “Olá”, usuário, ”,”, msg,”Você está logado!”
Fim


Programa testando_procedimento
Início
    Saudacao2(“Edson”, 16);
Fim
```

Código-fonte 8 – Procedimento “saudacao2” no Pseudocódigo

Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Criação do procedimento com parâmetros
def saudacao2(usuario, hora):
    if hora < 12:
        msg = "Bom dia!"
    elif hora < 18:
        msg = "Boa tarde!"
    else:
        msg = "Boa noite!"
    print("Olá "+ usuario +", "+ msg +" Você está logado")

# Chamada do procedimento
saudacao2("Edson", 16)
```

Código-fonte 9 – Procedimento “saudacao2” no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```java
import java.util.Scanner;
public class First {
        // Criação do Procedimento
        public static void saudacao2(String usuario, int hora)
        {
                String msg;
                if (hora < 12) {
                        msg = "Bom dia!";
                }else if (hora < 18) {
                        msg = "Boa tarde!";
                }else {
                        msg = "Boa noite!";
                }
                System.out.println("Ola "+ usuario + ", " + msg + " Você está logado.");


        }

        // Chamada do Procedimento
        public static void main(String[] args)
        {
                saudacao2("Edson", 16);
        }
```

```
}
```

Código-fonte 10 – Procedimento “saudacao2” no Java
Fonte: Elaborado pelo autor (2022)

4 FUNÇÕES SEM PARÂMETROS

Agora vamos falar de funções.

A mecânica de funcionamento de função e procedimento é quase idêntica, a diferença é que uma função **retorna um valor** do subalgoritmo (com o comando return) para o algoritmo que o chamou e este programa utiliza para algum fim este valor retornado.

Vejamos a ilustração de uma função que não passa parâmetro:

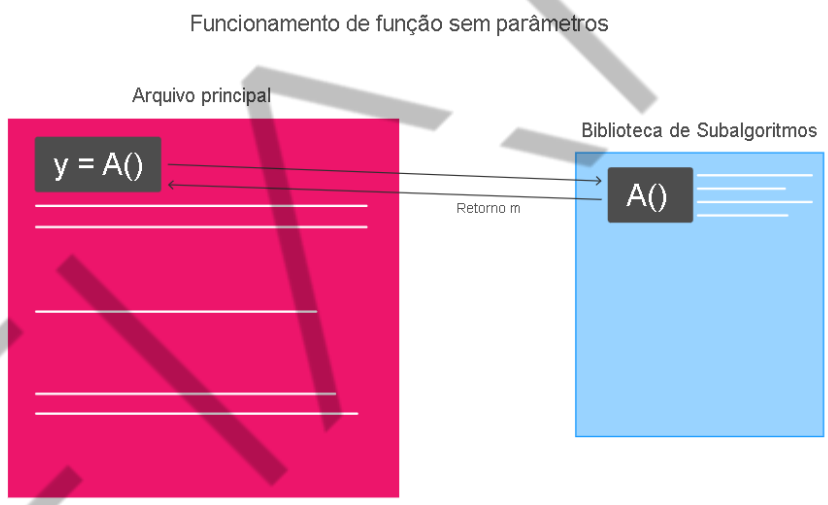

Figura 8– Ilustração do mecanismo de funcionamento da função sem parâmetros
Fonte: Elaborada pelo autor (2022)

O “Arquivo Principal” está invocando a função A() que está no arquivo “Biblioteca de Subalgoritmo” sem passar parâmetro. Executa-se a função na Biblioteca e, ao voltar ao programa principal, retorna a informação m e o fluxo permanece na mesma linha.

Por uma função retornar um valor, este valor deve ser manipulado no programa principal, seja atribuindo o seu retorno a uma variável ou acompanhado de algum comando. Caso você não faça isso, tem linguagem que retorna um erro de compilação (como a linguagem C), outras ignoram a linha e não fazem nada (como o Java).

As sintaxes em Pseudocódigo, Python e Java de função sem parâmetros são:

Momento Pseudocódigo

```
Função <nome da função>():<Tipo retorno>
Inicio
    <corpo da função>
Fim
```

Momento Python

```
def <nome da função>():
    corpo da função
    return(<valor>)
```

Momento Java

```
public static <tipo da função> <nome do procedimento>()
{
    <corpo do procedimento>;

}
```

Código-fonte 11 – Sintaxes de função sem parâmetro em Pseudocódigo, Python e Java
Fonte: Elaborado pelo autor (2022)

Vamos ver um exemplo de como utilizamos uma função sem parâmetros. Sabemos que o valor de pi (3,14159...) é um índice fundamental para calcularmos a área lateral de um círculo. Nem sempre lembraremos o valor de pi, então, vamos criar uma função que não passe parâmetro, mas retorne o valor de pi:

Criar uma função que não passe parâmetros e retorne o valor de pi

Execução do exemplo:

```
PI = 3.14159
```

Quadro 3 – Execução da função “pi” – sem parâmetros
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

```
Função pi(): Real
Início
    Retornar 3.14159;
Fim

Programa função
Início
    Escreva "PI = ", pi()
```

```
Fim
```

Código-fonte 12 – Função “pi” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Criação da função sem parâmetro
def pi():
    return 3.1415;

# Chamada da função
print ("PI = ", pi())
```

Código-fonte 13 – Função “pi” no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```java
import java.util.Scanner;
public class First {
        // Criação do Procedimento
        public static double pi()
        {
                return 3.14159;
        }
        // Chamada do Procedimento
        public static void main(String[] args)
        {
                System.out.println("PI = " + pi());
        }
}
```

Código-fonte 14 – Função “pi” no Java
Fonte: Elaborado pelo autor (2022)

5 FUNÇÕES COM PARÂMETROS

Como funcionam as funções que passam parâmetros? Esses tipos de funções são as mais usuais. Vejamos como elas funcionam.

Similar ao procedimento que passa parâmetro, o objetivo da função que passa parâmetro é o de transportar informações do algoritmo para o subalgoritmo e poder utilizar o parâmetro como variável dentro da função.

Vejamos a ilustração de uma função que passa parâmetro:

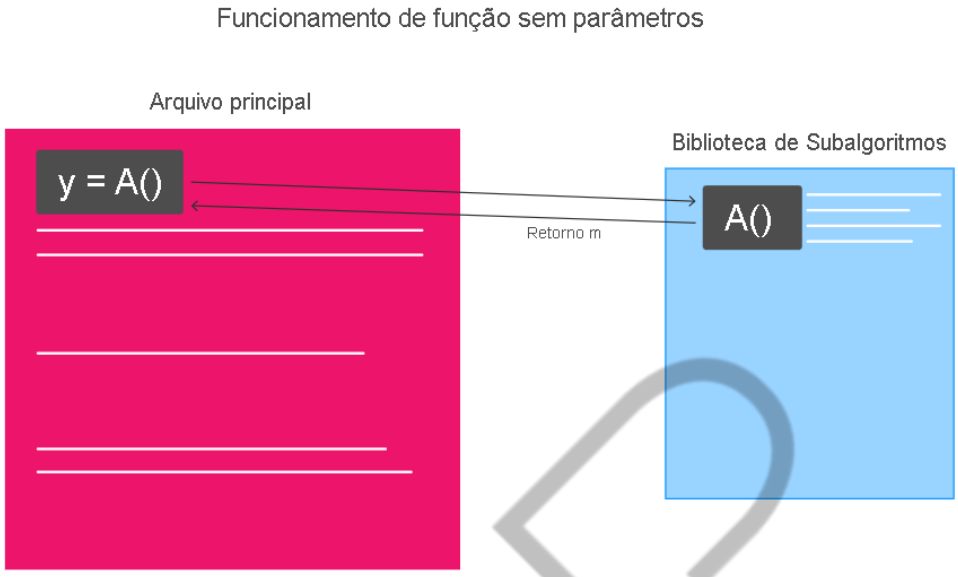

Figura 9 – Ilustração do mecanismo de funcionamento da função com parâmetros
Fonte: Elaborada pelo autor (2022)

O “Arquivo Principal” está invocando a função A(x) que está no arquivo “Biblioteca de Subalgoritmo” passando x como parâmetro. Executa-se a função na Biblioteca, utilizando x em algum momento e, ao voltar ao programa principal, RETORNA a informação m e o fluxo permanece na mesma linha.

As sintaxes em Pseudocódigo, Python e Java de função que passam parâmetros são:

Momento Pseudocódigo

```
Função <nome da função>([<parâmetros:tipos>]):<Tipo retorno>
Início
    <corpo da função>
Fim
```

Momento Python

```
def <nome da função>([<parâmetros>]):
    corpo da função
    return <valor>
```

Momento Java

```
public static <tipo função> <nome da função>(<tipo> <parâmetros>)
{
```

```
    <corpo da função>;
}
```

Código-fonte 15 – Sintaxes de função com parâmetro em Pseudocódigo, Python e Java
Fonte: Elaborado pelo autor (2022)

Dados os conceitos, vamos criar uma função que passe parâmetros e retorne um valor:

Criar uma função que retorne o maior entre dois números passados por parâmetro

Execução do exemplo:

```
Digite um número: 6
Digite outro número: 9

Maior número: 9
```

Quadro 4 – Execução da função “maior2n” – com parâmetros
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

```
Função maior2n(int num1, int num 2): inteiro
Var
   maior: inteiro
Início
   Se num1 > num2 então
       maior = num1
   senão
       maior = num2
   fim_se
   retornar (maior)
Fim


Programa testando_funcao
Var
    n1, n2: inteiro
Início
    Escreva “Digite um número:”
    Leia n1
    Escreva “Digite outro número:”
    Leia n2
    Escreva “Maior número: ”, maior2n(n1, n2)
Fim
```

Código-fonte 16 – Função “maior2n” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Criação da função com parâmetro
def maior2n(num1, num2):
    if num1 > num2:
        maior = num1
    else:
        maior = num2
    return maior

# Programa principal
n1 = int(input("Digite um número: "))
n2 = int(input("Digite outro número: "))
print("Maior numero: ", maior2n(n1,n2))
```

Código-fonte 17 – Função “maior2n” no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```java
import java.util.Scanner;
public class First {
    // Criação da função
    public static int maior2n(int num1, int num2)
    {
        int maior;
        if (num1 > num2) {
            maior = num1;
        }else {
            maior = num2;
        }
        return maior;
    }
    // Programa principal
    public static void main(String[] args)
    {
        Scanner teclado = new Scanner(System.in);
        int n1, n2;
        System.out.println("Digite um número: ");
        n1 = teclado.nextInt();
        System.out.println("Digite outro número: ");
        n2 = teclado.nextInt();
        System.out.println("Maior numero = " + maior2n(n1, n2));
    }
}
```

Código-fonte 18 – Função “maior2n” no Java
Fonte: Elaborado pelo autor (2022)

6 DE-PARA: PSEUDOCÓDIGO, PYTHON E JAVA – FUNÇÕES E PROCEDIMENTOS

Para fazermos o de-para de subalgoritmos para as linguagens que estamos trabalhando, vamos utilizar um algoritmo escrito de forma estruturada. Depois modificaremos esse algoritmo para conceito de Algoritmo | Subalgoritmo.

Considere que esse algoritmo é para a análise de apenas um aluno:

Em uma instituição de ensino, um aluno é submetido a três avaliações em um semestre. A média semestral é calculada por meio de uma média simples das duas maiores avaliações obtidas entre três avaliações.

Caso essa média semestral resulte em uma nota inferior a 4, o aluno foi reprovado sem outra oportunidade.

Caso a média semestral seja maior ou igual a 7, o aluno foi aprovado de forma direta.

Caso a média esteja entre 4 e 6.9, o aluno tem a oportunidade de fazer o exame por meio de uma nova avaliação.

Considerando que o aluno está em exame, a média final é uma média simples da média semestral com a nota da avaliação obtida no exame. Caso a média final seja inferior a 5, o aluno foi Reprovado em Exame, senão ele foi aprovado.

Requisitos:

- O algoritmo efetua todo esse cálculo com apenas um aluno.

- Consistir as notas para que estejam entre 0 e 10.

- As mensagens informativas devem ser claras em relação ao problema ou à situação do aluno.

- Quando necessário, exibir as médias calculadas para simples conferência.

Execução do exemplo:

```
Nota 1:5
```

```
Nota 2:6
Nota 3:7
A sua média semestral é 6.5
VOCÊ FICOU EM EXAME
6
Aprovado em exame com media 6.25
```

Quadro 5 – Execução do exemplo “Média semestral com exame”
Fonte: Elaborado pelo autor (2022)

Usaremos as resoluções estruturadas abaixo como base para criarmos os subalgoritmos:

Momento Pseudocódigo

```
Programa media_aluno
  nota1, nota2, nota3, media_semestral: real
  nota_exame, media_exame: real
Início
  Escreva “Nota 1:”
  Leia nota1
  Se nota1 >= 0 e nota1 <= 10 então
    Leia nota2
    Se nota2 >= 0 e nota2 <= 10 então
      Leia nota3
      Se nota3>=0 e nota3<=10 então
        menor_nota = nota1
        Se nota2 < menor_nota então
          menor_nota = nota2
        fim_se
        Se nota3 < menor_nota então
          menor_nota = nota3
        fim_se
        media_semestral = (nota1 + nota2 + nota3 – menor_nota)/2
        Escreva “A sua média semestral é”, media_semestral
        Se media_semestral < 4 então
          Escreva “Você está reprovado direto”
        Senão se media_semestral >= 7 então
          Escreva “Você está aprovado direto”
        Senão
          Escreva “VOCÊ FICOU EM EXAME”
          Leia nota_exame
          Se nota_exame >= 0 e nota_exame <= 10 então
            media_exame = (media_semestral + nota_exame) / 2
            Se media_exame < 5 então
              Escreva “Reprovado em exame com media”,media_exame
            Senão
              Escreva “Aprovado em exame com media”, media_exame
            Fim_se
          Senão
            Escreva “Nota de exame”, nota_exame, “Inválida”
```

```
            Fim_se
          Fim_se
        Senão
          Escreva "Nota 3", nota3, "- É inválida"
        Fim_Se
      Senão
        Escreva "Nota 2", nota2, "- É inválida"
      Fim_se
    Senão
      Escreva "Nota 1", nota1, "- É inválida"
    Fim_se
Fim
```

Código-fonte 19 – Exemplo "Média semestral com exame" escrito de forma estruturada no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Leitura da Nota
nota1 = float(input("Nota 1: "))
# Verificação se a nota é válida
if nota1 >=0 and nota1  <= 10:
   nota2 = float(input("Nota 2: "))
   if nota2 >= 0 and nota2 <= 10:
      nota3 = float(input("Nota 3: "))
      if nota3 >= 0 and nota3 <= 10:
         # Verificação da menor entre 3 notas
         menor_nota = nota1
         if nota2 < menor_nota:
            menor_nota = nota2
         if nota3 < menor_nota:
            menor_nota = nota3
         # Cálculo da media semestral com as duas notas maiores
         media_semestral = (nota1 + nota2 + nota3 - menor_nota) / 2
         print(f"A sua Média Semestral é {media_semestral:.1f}")
         # Verificação do Status de aprovação ou não do aluno
         if media_semestral < 4:
            print("Você está Reprovado direto.")
         elif media_semestral >= 7:
            print("Voce está Aprovado direto.")
         else:
            # Caso o aluno tenha ficado em exame
            print("\nVOCÊ FICOU EM EXAME!\n")
            nota_exame = float(input("Digite a nota do exame: "))
            if nota_exame >= 0 and nota_exame <= 10:
               media_exame = (media_semestral + nota_exame) / 2
               # Status de aprovação ou não do aluno após o exame
               if media_exame < 5:
                  print(f"\nReprovado em exame com média {media_exame:.1f}")
               else:
                  print(f"\nAprovado em exame com média {media_exame:.1f}")
            else:
```

```
                print(f"Nota de exame {nota_exame} inválida!")
        else:
            print(f"Nota 3: {nota3} - É inválida!")
    else:
        print(f"Nota 2: {nota2} - É inválida!")
else:
    print (f"Nota 1: {nota1} - É inválida!")
```

Código-fonte 20 – Exemplo “Média semestral com exame” escrito de forma estruturada no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```
import java.util.Scanner;
public class First {
	public static void main(String[] args)
	{
		// Estancia o objeto Teclado para ler variáveis
		Scanner teclado = new Scanner(System.in);
		// Declaração das variáveis
		float nota1, nota2, nota3, media_semestral, nota_exame, media_exame, menor_nota;
		System.out.print("Nota 1:");
		nota1 = teclado.nextFloat();
		// Verifica se a nota é válida
		if (nota1 >= 0 && nota1 <= 10)
		{
			System.out.print("Nota 2:");
			nota2 = teclado.nextFloat();
			if (nota2 >= 0 && nota2 <= 10)
			{
				System.out.print("Nota 3:");
				nota3 = teclado.nextFloat();
				if (nota3 >= 0 && nota3 <= 10)
				{
					// Verifica qual a menor nota
					menor_nota = nota1;
					if (nota2 < menor_nota)
					{
						menor_nota = nota2;
					}
					if (nota3 < menor_nota)
					{
						menor_nota = nota3;
					}
					// Calcula a média tirando a nota com menor valor
				media_semestral = (nota1 + nota2 + nota3 - menor_nota) / 2;
				System.out.println("A sua média semestral é " + media_semestral);
			// Verifica o status do aluno
					if (media_semestral < 4)
					{
						System.out.println("Você está reprovado direto");
					}
```

```
                    else if (media_semestral >= 7)
                      {
                            System.out.println("Você está aprovado direto");
                      }
                      else
                      {
                      // Solicita uma nota em caso de exame
                            System.out.println("VOCÊ FICOU EM EXAME");
                            nota_exame = teclado.nextFloat();
                            if(nota_exame >= 0 && nota_exame <= 10)
                            {
                                  // Verifica se passou no exame
                                  media_exame = (media_semestral + nota_exame) / 2;
                                  if(media_exame < 5)
                                  {
                                        System.out.println("Reprovado em exame com media " +
media_exame);
                                  }
                                  else
                                  {
                                        System.out.println("Aprovado em exame com media " +
media_exame);
                                  }
                            }
                            else
                            {
                                  System.out.println("Nota de exame" + nota_exame + "Inválida");
                            }
                      }
                }
                else
                {
                      System.out.println("Nota 3 " + nota3 + " - É inválida");
                }
          }
          else
          {
                System.out.println("Nota 2 " + nota2 + " - É inválida");
          }
     }
     else
     {
          System.out.println("Nota 1 " + nota1 + " - É inválida");
     }
  }
}
```

Código-fonte 21 – Exemplo “Média semestral com exame” escrito de forma estruturada no Java
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

Faremos a decomposição do programa estruturado no Pseudocódigo utilizando funções e procedimentos. Inicialmente apresentaremos cada subalgoritmo e no final colocaremos o código-fonte completo.

Este primeiro subalgoritmo é uma função do tipo booleana. Ela pega um parâmetro do tipo real que representa uma nota e retorna verdade caso seja uma nota válida ou falso caso não seja:

```
// Função que verifica se uma nota é válida
Função nota_valida(nota: real): Lógica
Início
     Se (nota >= 0 e nota <= 10)então
          retorne verdade
     Então
          retorne falso
     Fim_se
Fim
```

Código-fonte 22 – Função “nota_valida” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

O segundo subalgoritmo é uma função do tipo real que analisa três números passados por parâmetro e retorna o de menor valor:

```
// Função que retorna o menor entre 3 valores
Função menor3n(n1: real, n2: real, n3:real): Real
Var
     menor: real
Início
     // Verifica qual a menor nota
     menor = n1
     se (n2 < menor) então
          menor = n2
     fim_se
     se (n3 < menor) então
          menor = n3
     fim_se
     retorne menor
Fim
```

Código-fonte 23 – Função “nota_valida” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

O terceiro subalgoritmo tem uma característica interessante: ele utiliza dentro dele a chamada de outra função “menor3n”. Lembra que no enunciado do exemplo dizia: *“*A média semestral é calculada por meio de uma média simples das duas maiores notas obtidas”? Então, essa função chama a função que retorna o menor entre três valores antes de calcular a média (tirando a de menor valor):

```
// Função que retorna a média de 3 números
função media2maiores(n1: real, n2: real, n3: real): real
var
    menor: real
início
     chamada de uma função dentro de outra
     menor = menor3n(n1,n2,n3)
     retorne (n1 + n2 + n3 - menor) / 2;
Fim
```

Código-fonte 24 – Função “media2maiores” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

O próximo subalgoritmo é um procedimento que simplesmente pega uma média passada por parâmetro e retorna uma mensagem incorporando a média:

```
// Procedimento que exibe a mensagem da media semestral
Procedimento msg_media_semestral(m: real)
Início
     Escreva "A sua média semestral é ", m
Fim
```

Código-fonte 25 – Procedimento “msg_media_semestral” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

A função abaixo pega dois números reais passados por parâmetro e retorna a média obtida:

```
// Função que calcula a média de dois números
Função media2n(n1: real, n2: real): real
Início
     retorne (n1 + n2) / 2
Fim
```

Código-fonte 26 – Função “media2n” no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Diferentemente da mensagem do último procedimento, decidimos fazer a mensagem ser retornada em forma de função. Essa função pega a média de exame passada por parâmetro e retorna um texto composto por uma combinação de texto (string) e um valor real que está no parâmetro:

```
// Função que retorna mensagem de aprovado ou não no exame
Função msg_aprovado_exame(float m): Texto
Início
     Se (m < 5) então
          retorne ("Reprovado em exame com media " + m)
     else

          retorne ("Aprovado em exame com media " + m)
     Fim_se
Fim
```

Código-fonte 27 – Função “msg_aprovado_exame” no Pseudocódigo

Fonte: Elaborado pelo autor (2022)

Depois de comentados todos os subalgoritmos, vamos agora colocar o programa na íntegra: com os subalgoritmos e o programa principal (algoritmo) juntos. Perceba como fica mais dinâmico. Veja as chamadas dos subalgoritmos comentados dentro do algoritmo:

```
// Função que verifica se uma nota é válida
Função nota_valida(nota: real):Lógica
Início
     se (nota >= 0 e nota <= 10)então
          returnar verdade
     senão
          returnar falso
     fim_se
fim

// Função que retorna o menor entre 3 valores
Função menor3n(n1: real, n2: real, n3:real)
Var
     menor: real
inicio
     // Verifica qual a menor nota
     menor = n1
     se (n2 < menor) então
          menor = n2
     fim_se
     se (n3 < menor) então
          menor = n3
     fim_se
     retornar menor
fim

// Função que retorna a média de 3 números
função media2maiores(n1: real, n2: real, n3: real): real
var
    menor: real
início
     chamada de uma função dentro de outra
     menor = menor3n(n1,n2,n3)
     retornar (n1 + n2 + n3 - menor) / 2;
Fim

// Procedimento que exibe a mensagem da media semestral
Procedimento msg_media_semestral(m: real)
Início
     Escreva "A sua média semestral é ", m
Fim

// Função que retorna mensagem de aprovado ou não no exame
```

```
Função msg_aprovado_exame(float m): Texto
Início
     Se (m < 5) então
          retornar ("Reprovado em exame com media " + m)
     Senão

          retornar ("Aprovado em exame com media " + m)
     Fim_se
Fim

// PROGRAMA PRINCIPAL
Programa Principal
   nota1, nota2, nota3, media_semestral, nota_exame, media_exame,
menor_nota: real
Início
   Escreva "Nota 1:"
   Leia nota1

   // chamada da função 'nota_valida'
   Se (nota_valida(nota1)) então
       Escreva "Nota 2:"
       Leia nota2
       Se(nota_valida(nota2))
          Escreva "Nota 3:"
          Leia nota3

          // chamada da função nota válida
          Se (nota_valida(nota3))então

              // Chamada da função 'menor3n'
              menor_nota = menor3n(nota1, nota2, nota3)

              // chamada da função 'media2maiores' que calcula a media
              // descartando a menor
              media_semestral = media2maiores(nota1, nota2, nota3

              // chamada do procedimento que exibe a mensagem da media
              // semestral
              msg_media_semestral(media_semestral)

              // Verifica o status do aluno
              Se (media_semestral < 4) então
                  Escreva "Você está reprovado direto"
              Senão se (media_semestral >= 7) então
                  Escreva "Você está aprovado direto")
              Senão
                  // Solicita uma nota em caso de exame
                  Escreva "VOCÊ FICOU EM EXAME"
                  Leia nota_exame
                  Se(nota_valida(nota_exame)) então
                      // chamada da função media_exame que calcula a media
                      media_exame = media2n(media_semestral, nota_exame)

                      // chamada da função msg_aprovado_exame
                      Escreva msg_aprovado_exame(media_exame
                  Senão
                      Escreva "Nota de exame" + nota_exame + "Inválida"
                  Fim_se
              Fim_se
          Senão
```

```
                Escreva "Nota 3 " + nota3 + " - É inválida"
            Fim_se
        Senão
            Escreva "Nota 2 " + nota2 + " - É inválida"
    Senão
        Escreva "Nota 1 " + nota1 + " - É inválida"
    Fim_Se
Fim
```

Código-fonte 28 – Exemplo “Média semestral com exame” escrito com o conceito de subalgoritmos no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

Faremos a decomposição do programa estruturado em Python utilizando funções e procedimentos. Apresentaremos cada subalgoritmo e no final colocaremos o código-fonte completo.

Este primeiro subalgoritmo é uma função do tipo booleana. Ela pega um parâmetro do tipo float que representa uma nota e retorna True caso seja uma nota válida ou False caso não seja:

```
# Função booleana que verifica se uma nota é válida ou não
def nota_valida(nota):
    if nota >= 0 and nota <= 10:
        return True
    else:
        return False
```

Código-fonte 29 – Função “nota_valida” no Python
Fonte: Elaborado pelo autor (2022)

O segundo subalgoritmo é uma função do tipo real que analisa três números passados por parâmetro e retorna o de menor valor:

```
/ # Função que retorna o menor entre 3 valores
def menor3n(n1, n2, n3):
    menor = n1
    if n2 < menor:
        menor = n2
    if n3 < menor:
        menor = n3
    return menor
```

Código-fonte 30 – Função “maior3n” no Python
Fonte: Elaborado pelo autor (2022)

O terceiro subalgoritmo tem uma característica interessante: ele utiliza dentro dele a chamada de outra função “menor3n”. Lembra que no enunciado do exemplo dizia: *“A média semestral é calculada por meio de uma média simples das duas maiores notas obtidas”*? Então, essa função chama a função que retorna o menor entre três valores antes de calcular a média (tirando a de menor valor):

```
# Função que calcula a media das 2 maiores notas
def media2maiores(n1, n2, n3):
    menor = menor3n(n1, n2, n3)
    return (n1 + n2 + n3 - menor) / 2
```

Código-fonte 31 – Função “media2maiores” no Python
Fonte: Elaborado pelo autor (2022)

O próximo subalgoritmo é um procedimento que simplesmente pega uma média passada por parâmetro e retorna uma mensagem incorporando a média:

```
# Procedimento para exibir a média semestral
def msg_media_semestral(m):
    print(f"A sua Média Semestral é {m:.1f}")
```

Código-fonte 32 – Procedimento “msg_media_semestral” no Python
Fonte: Elaborado pelo autor (2022)

A função abaixo pega dois números float passados por parâmetro e retorna a média obtida:

```
# Função que calcula a média de 2 números
def media2n(n1, n2):
    return (n1 + n2) / 2
```

Código-fonte 33 – Função “media2n” no Python
Fonte: Elaborado pelo autor (2022)

Diferentemente da mensagem do último procedimento, decidimos fazer a mensagem ser retornada em forma de função. Essa função pega a média de exame passada por parâmetro e retorna um texto composto por uma combinação de texto (string) e um valor real que está no parâmetro:

```
# Função que retorna uma msg de aprovado ou nao no exame
def msg_aprovado_exame(m):
    if m < 5:
        return "Reprovado em exame com média " + str(m)
    else:
        return "Aprovado em exame com média " + str(m)
```

Código-fonte 34 – Função “msg_aprovado_exame” no Python

Fonte: Elaborado pelo autor (2022)

Depois de comentados todos os subalgoritmos, vamos agora colocar o programa na íntegra: com os subalgoritmos e o programa principal (algoritmo) juntos. Perceba como fica mais dinâmico. Veja as chamadas dos subalgoritmos comentados dentro do programa principal:

```
# Função booleana que verifica se uma nota é válida ou não
def nota_valida(nota):
    if nota >= 0 and nota <= 10:
        return True
    else:
        return False


# Função que retorna o menor entre 3 valores
def menor3n(n1, n2, n3):
    menor = n1
    if n2 < menor:
        menor = n2
    if n3 < menor:
        menor = n3
    return menor


# Função que calcula a media das 2 maiores notas
def media2maiores(n1, n2, n3):
    menor = menor3n(n1, n2, n3)
    return (n1 + n2 + n3 - menor) / 2


# Procedimento para exibir a média semestral
def msg_media_semestral(m):
    print(f"A sua Média Semestral é {m:.1f}")


# Função que calcula a média de 2 números
def media2n(n1, n2):
    return (n1 + n2) / 2


# Função que retorna uma msg de aprovado ou nao no exame
def msg_aprovado_exame(m):
    if m < 5:
        return "Reprovado em exame com média " + str(m)
```

```
    else:
        return "Aprovado em exame com média " + str(m)


# Leitura da Nota
nota1 = float(input("Nota 1: "))
# Verificação se a nota é válida
if nota_valida(nota1):
nota2 = float(input("Nota 2: "))
    if nota_valida(nota2):
        nota3 = float(input("Nota 3: "))
        if nota_valida(nota3):
            # Cálculo da media semestral com as duas notas maiores
            media_semestral = media2maiores(nota1, nota2, nota3)
            msg_media_semestral(media_semestral)
            # Verificação do Status de aprovação ou não do aluno
            if media_semestral < 4:
                print("Você está Reprovado direto.")
            elif media_semestral >= 7:
                print("Voce está Aprovado direto.")
            else:
                # Caso o aluno tenha ficado em exame
                print("\nVOCÊ FICOU EM EXAME!\n")
                nota_exame = float(input("Digite a nota do exame: "))
                if nota_valida(nota_exame)::
                    media_exame = media2n(media_semestral, nota_exame)
                    # Status de aprovação ou não do aluno após o exame
                    print(msg_aprovado_exame(media_exame))
                else:
                    print(f"Nota de exame {nota_exame} inválida!")
        else:
            print(f"Nota 3: {nota3} - É inválida!")
    else:
        print(f"Nota 2: {nota2} - É inválida!")
else:
    print(f"Nota 1: {nota1} - É inválida!")
```

Código-fonte 35 – Exemplo “Média semestral com exame” escrito com o conceito de subalgoritmos no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

Faremos a decomposição do programa estruturado em Java utilizando funções e procedimentos. Apresentaremos cada subalgoritmo e no final colocaremos o código-fonte completo.

Este primeiro subalgoritmo é uma função do tipo booleana. Ela pega um parâmetro do tipo float que representa uma nota e retorna True caso seja uma nota válida ou False caso não seja:

```
// Função que verifica se uma nota é válida
public static boolean nota_valida(float nota){
        if (nota >= 0 && nota <= 10){
                return true;
        }else{
                return false;
        }
}
```

Código-fonte 36 – Função “nota_valida” no Java
Fonte: Elaborado pelo autor (2022)

O segundo subalgoritmo é uma função do tipo float que analisa três números passados por parâmetro e retorna o de menor valor:

```
// Função que retorna o menor entre 3 valores
public static float menor3n(float n1, float n2, float n3){
        // Verifica qual a menor nota
        float menor = n1;
        if (n2 < menor)
        {
                menor = n2;
        }
        if (n3 < menor)
        {
                menor = n3;
        }
        return (menor);
}
```

Código-fonte 37 – Função “maior3n” no Java
Fonte: Elaborado pelo autor (2022)

O terceiro subalgoritmo tem uma característica interessante: ele utiliza dentro dele a chamada de outra função “menor3n”. Lembra que no enunciado do exemplo dizia: *“A média semestral é calculada por meio de uma média simples das duas maiores notas obtidas”*? Então, essa função chama a função que retorna o menor entre três valores antes de calcular a média (tirando a de menor valor):

```
// Função que retorna a média de 3 números
private static float media2maiores(float n1, float n2, float n3){
        float menor = menor3n(n1,n2,n3);
        return (n1 + n2 + n3 - menor) / 2;
}
```

Código-fonte 38 – Função “media2maiores” no Java
Fonte: Elaborado pelo autor (2022)

O próximo subalgoritmo é um procedimento que simplesmente pega uma média passada por parâmetro e retorna uma mensagem incorporando a média:

```
// Procedimento que exibe a mensagem da media semestral
private static void msg_media_semestral(float m) {
        System.out.println("A sua média semestral é " + m);
}
```

Código-fonte 39 – Procedimento “msg_media_semestral” no Java
Fonte: Elaborado pelo autor (2022)

A função abaixo pega dois números float passados por parâmetro e retorna a média obtida:

```
// Função que calcula a média de dois números
public static float media2n(float n1, float n2) {
        return (n1 + n2) / 2;
}
```

Código-fonte 40 – Função “media2n” no Java
Fonte: Elaborado pelo autor (2022)

Diferentemente da mensagem do último procedimento, decidimos fazer a mensagem ser retornada em forma de função. Essa função pega a média de exame passada por parâmetro e retorna um texto composto por uma combinação de texto (string) e um valor real que está no parâmetro:

```
// Função que retorna mensagem de aprovado ou não no exame
public static String msg_aprovado_exame(float m) {
        if(m < 5)
        {
                return ("Reprovado em exame com media " + m);
        }
        else
        {
                return ("Aprovado em exame com media " + m);
        }
}
```

Código-fonte 41 – Função “msg_aprovado_exame” no Java
Fonte: Elaborado pelo autor (2022)

Depois de comentados todos os subalgoritmos, vamos agora colocar o programa na íntegra: com os subalgoritmos e o programa principal (algoritmo) juntos. Perceba como fica mais dinâmico. Veja as chamadas dos subalgoritmos comentados dentro do programa principal (main):

```
import java.util.Scanner;
public class First {

	// Função que verifica se ma nota é válida
	public static boolean nota_valida(float nota){
		if (nota >= 0 && nota <= 10){
			return true;
		}else{
			return false;
		}
	}

	// Função que retorna o menor entre 3 valores
	public static float menor3n(float n1, float n2, float n3) {
		// Verifica qual a menor nota
		float menor = n1;
		if (n2 < menor)
		{
			menor = n2;
		}
		if (n3 < menor)
		{
			menor = n3;
		}
		return (menor);
	}

	// Função que retorna a média de 3 números
	private static float media2maiores(float n1, float n2, float n3) {
		float menor = menor3n(n1,n2,n3);
		return (n1 + n2 + n3 - menor) / 2;
	}

	// Procedimento que exibe a mensagem da media semestral
	private static void msg_media_semestral(float m) {
		System.out.println("A sua média semestral é " + m);
	}

	// Função que calcula a média de dois números
	public static float media2n(float n1, float n2) {
		return (n1 + n2) / 2;
```

```
    }

    // Função que retorna mensagem de aprovado ou não no exame
    public static String msg_aprovado_exame(float m) {
        if(m < 5)
        {
            return ("Reprovado em exame com media " + m);
        }
        else
        {
            return ("Aprovado em exame com media " + m);
        }
    }

    public static void main(String[] args)
    {
        // Estancia o objeto Teclado para ler variáveis
        Scanner teclado = new Scanner(System.in);

        // Declaração das variáveis
        float nota1, nota2, nota3, media_semestral, nota_exame, media_exame, menor_nota;

        System.out.print("Nota 1:");
        nota1 = teclado.nextFloat();
        // chamada da função 'nota_valida'
        if (nota_valida(nota1))
        {
            System.out.print("Nota 2:");
            nota2 = teclado.nextFloat();
            if (nota_valida(nota2))
            {
                System.out.print("Nota 3:");
                nota3 = teclado.nextFloat();
                if (nota_valida(nota3))
                {
                    // Chamada da função 'menor3n'
                    menor_nota = menor3n(nota1, nota2, nota3);

                    // chamada da função 'media2maiores' que calcula a media descartando a
menor
                    media_semestral = media2maiores(nota1, nota2, nota3);

                    // chamada do procedimento que exibe a mensagem da media semestral
                    msg_media_semestral(media_semestral);

                // Verifica o status do aluno
                    if (media_semestral < 4)
                    {
                        System.out.println("Você está reprovado direto");
                    }
                else if (media_semestral >= 7)
                    {
```

```
                        System.out.println("Você está aprovado direto");
                    }
                    else
                    {

                        // Solicita uma nota em caso de exame
                        System.out.println("VOCÊ FICOU EM EXAME");
                        nota_exame = teclado.nextFloat();
                        if(nota_valida(nota_exame))
                        {

                            // chamada da função media_exame que calcula a media
                            media_exame = media2n(media_semestral, nota_exame);

                            // chamada da função msg_aprovado_exame
                            System.out.println(msg_aprovado_exame(media_exame));
                        }
                        else
                        {
                            System.out.println("Nota de exame" + nota_exame + "Inválida");
                        }
                    }
                }
                else
                {
                    System.out.println("Nota 3 " + nota3 + " - É inválida");
                }
            }
            else
            {
                System.out.println("Nota 2 " + nota2 + " - É inválida");
            }
        }
        else
        {
            System.out.println("Nota 1 " + nota1 + " - É inválida");
        }
    }
}
```

Código-fonte 42 – Exemplo “Média semestral com exame” escrito com o conceito de subalgoritmos no Java
Fonte: Elaborado pelo autor (2022)

7 VETORES

No primeiro capítulo, aprendemos a utilizar variáveis de memória. Nela conseguimos armazenar uma informação por variável e, se mudarmos o conteúdo da variável, ela sobrepõe o valor anterior, permanecendo sempre a última informação. Então perguntamos: “E se precisarmos manter o histórico dos conteúdos da variável?”

Com uma variável de memória comum não conseguiremos, apenas se criarmos várias variáveis, mas com o conceito de variáveis indexadas conseguimos guardar vários valores na mesma variável. É sobre ela que falaremos agora.

Variáveis indexadas são aquelas em que podemos guardar várias informações e a sua manipulação é feita pelo índice, que fica entre colchetes. O índice sempre inicia do 0 (zero) e vai até o limite do vetor. Temos a variável indexada unidimensional (uma linha e diversas colunas), que é o vetor, e a bidimensional (diversas linhas e diversas colunas), que é a matriz. Neste texto falaremos da primeira.

As variáveis indexadas são os conhecidos vetores e matrizes; elas também são conhecidas como listas ou até tuplas (vetor de constantes).

Como variáveis de memória, um vetor deve ter um nome e um tipo. A mudança fica na definição do tamanho do vetor, ou seja, a quantidade de elementos que ele terá. Vejamos um comparativo de declaração de uma variável comum com um vetor declarados no Pseudocódigo:

```
Var
    // Declaração de uma variável comum x do tipo inteiro
    x: inteiro
    // Declaração de um vetor de inteiros com dez posições
    y[10]: inteiro
```

Código-fonte 43 – Exemplo de declaração de vetor no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Cada linguagem de programação tem a sua sintaxe e até uma forma diferente de tratar o vetor. Veremos cada uma (no Pseudocódigo, Python e Java) no decorrer do capítulo.

Vejamos agora uma ilustração do vetor declarado acima:

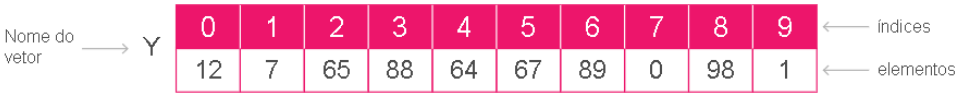

Figura 10 – Ilustração dos componentes de um vetor
Fonte: Elaborada pelo autor (2022)

Repare que o índice do vetor sempre começa com 0 (zero). Assim, o terceiro elemento tem o índice 2, por exemplo.

A manipulação do vetor funciona da seguinte forma:

```
// Atribui o valor 65 no índice 2 do vetor
y[2] = 65
// permite ao usuário digitar o elemento com índice 4
Leia y[4]
// Exibe o elemento do índice 0
Escreva "Primeira posição do vetor", y[0]
// Efetua cálculos e atribuições
Y[3] = y[6] + y[7]
// Verifica se um elemento é positivo
Se y[5] > 0 então
    Escreva "É positivo"
Senão
    Escreva "Não é positivo"
Fim_se
```

Código-fonte 44 – Manipulação de vetor no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Com este exemplo, vimos que um vetor pode ser tratado da mesma forma que uma variável comum em todos os comandos, devemos apenas nos lembrar de colocar o índice entre colchetes.

Vejamos como se declaram vetores no Pseudocódigo, Python e Java:

Momento Pseudocódigo

```
vetor_exemplo[10] : inteiro
```

Momento Python

```
vetor_exemplo = []
```

Momento Java

```
int [] vetor_exemplo = new int[10];
```

Código-fonte 45 – Declaração de vetor no Pseudocódigo, Python e Java
Fonte: Elaborado pelo autor (2022)

8 DE-PARA: VETOR

Para darmos exemplos sobre o conceito de vetor, vamos sugerir alguns problemas que serão resolvidos no Pseudocódigo, Python e Java:

Em um vetor de inteiros de 10 posições, fazer as seguintes rotinas:

1 – Preencher o vetor

2 – Exibir o conteúdo do vetor

3 – Somar os elementos do vetor

4 – Buscar um elemento no vetor

Veja a execução das rotinas:

```
Digite a posição: vetor[0]= 5
Digite a posição: vetor[1]= 8
Digite a posição: vetor[2]= 78
Digite a posição: vetor[3]= 6
Digite a posição: vetor[4]= 2
Digite a posição: vetor[5]= 3
Digite a posição: vetor[6]= -8
Digite a posição: vetor[7]= 2
Digite a posição: vetor[8]= 3
Digite a posição: vetor[9]= 4
vetor[0]= 5
vetor[1]= 8
vetor[2]= 78
vetor[3]= 6
vetor[4]= 2
vetor[5]= 3
vetor[6]= -8
vetor[7]= 2
vetor[8]= 3
vetor[9]= 4
Somatória do vetor = 103
Digite o elemento: 2
        Elemento 2 encontrado no vetor
```

Quadro 6 – Execução das rotinas com vetor
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

Segue a codificação das rotinas em Pseudocódigo:

```
Programa manipulando_vetor
var
    vetor[10]: int
    achou : lógica
    i, soma, elem: inteiro
Início
    // 1 - PREENCHER O VETOR
    Para i de 0 até 10 inc 1 faça
        Escreva "Digite vetor[",i , "]= "
        Leia vetor[i]
    Fim_para

    // 2 - EXIBIR O CONTEÚDO DO VETOR
    Para i de 0 até 10 inc 1 faça
        Escreva "vetor[", i ,"]= ", vetor[i]);
    Fim_apra

    // 3 - SOMAR OS ELEMENTOS DO VETOR
    Para i de 0 até 10 inc 1 faça
        Escreva "vetor[", i , "]= ", vetor[i]);
    Fim_para

    // 4 - BUSCAR UM ELEMENTO NO VETOR
    achou = Falso;

    // Digitação do elemento que será procurado
    Escreva "Digite o elemento: "
    Leia elem

    Para i de 0 até 10 inc 1 faça
        // Caso encontre o elemento, interrompe a busca
        Se vetor[i] == elem então
            achou = Verdade
            i = 10
        Fim_se
    Fim_para

    // Analisa se encontrou ou não o elemento
    Se (achou) então
        Escreva "Elemento ", elem ," encontrado no vetor"
    senão
        Escreva "Elemento ", elem ," NÃO encontrado no vetor"
    Fim_se
Fim
```

Código-fonte 46 – Rotinas utilizando vetor no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

Python, por ser uma linguagem mais dinâmica, trata o vetor (lista) de forma diferente das demais linguagens. Não é definido o tamanho da lista inicialmente. Usamos os elementos de acordo com a necessidade. Sendo assim, existem comandos específicos para tratar a lista (enquanto nas outras linguagens a lista é tratada diretamente). Vejamos alguns comandos de listas em Python:

- O conteúdo dos elementos é heterogêneo, ou seja, cada célula pode ser de um tipo diferente (diferentemente das outras linguagens).
- Os elementos são dinâmicos, ou seja, acrescentamos e excluímos quantos elementos quisermos.
- O sinal de + concatena duas listas.
- O comando append acrescenta um item no final da lista:
  - lista.append(45) # acrescenta o elemento 45.
- O insert permite editar um elemento:
  - lista.insert(3,”Edson”) # coloca Edson no elemento representado pelo índice 3.
- O extended acrescenta uma lista no final da outra:
  - l1.extend(l2) # acrescenta a lista l2 no final da l1.
- O min() e o max() retornam o menor e o maior item da lista, respectivamente:
  - print(min(l3),max(l3)) # exibe o menor e o maior item da lista l3.
- O pop remove o último elemento da lista:
  - lista.pop() # remove o último elemento da lista.
- O del exclui um ou mais elementos da lista:
  - del(lista[3]) # remove o elemento com o índice 3.
- O clear apaga todos os elementos da lista.
  - lista.clear()

Dados estes conceitos, vejamos como ficou a codificação das rotinas em Python:

```
vetor = [];

# 1 - PREENCHER O VETOR
```

```
for i in range(0, 10, 1):
    print(f"Digite a posição: vetor[{i}]= ")
    elem = int(input())
    vetor.append(elem)

# 2 - EXIBIR O CONTEÚDO DO VETOR
for i in range(0, 10, 1):
    print(f"vetor[{i}]= {vetor[i]}")

# 3 - SOMAR OS ELEMENTOS DO VETOR
soma = 0
for i in range(0, 10, 1):
    soma += vetor[i];

print (f"Somatória do vetor = {soma}")

# 4 - BUSCAR UM ELEMENTO NO VETOR
achou = False

# Digitação do elemento que será procurado
elem = int(input("Digite o elemento:"))
for i in range(0, 10, 1):
    # Caso encontre o elemento, interrompe a busca
    if vetor[i] == elem:
        achou = True
        break

# Analisa se encontrou ou não o elemento
if (achou):
    print(f"Elemento {elem} encontrado no vetor")
else:
    print(f"Elemento {elem} NÃO  foi encontrado no vetor")
```

Código-fonte 47 – Rotinas utilizando vetor no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

Segue a codificação das rotinas em Java:

```
import java.util.Scanner;
public class First {

        public static void main(String[] args)
        {
```

```
// Estancia o objeto Teclado para ler variáveis
Scanner teclado = new Scanner(System.in);

int [] vetor = new int[10];

// 1 - PREENCHER O VETOR
for(int i = 0; i < 10; i++)
{
	System.out.print("Digite vetor["+ i + "]= ");
	vetor[i] = teclado.nextInt();
}

// 2 - EXIBIR O CONTEÚDO DO VETOR
for(int i = 0; i < 10; i++)
{
	System.out.println("vetor["+ i + "]= " + vetor[i]);
}

// 3 - SOMAR OS ELEMENTOS DO VETOR
for(int i = 0; i < 10; i++)
{
	System.out.println("vetor["+ i + "]= " + vetor[i]);
}

// 4 - BUSCAR UM ELEMENTO NO VETOR
boolean achou = false;
int elem;

// Digitação do elemento que será procurado
System.out.print("Digite o elemento: ");
elem = teclado.nextInt();

for(int i = 0; i < 10; i++)
{
	// Caso encontre o elemento, interrompe a busca
	if (vetor[i] == elem)
	{
		achou = true;
		break;
	}

}
// Analisa se encontrou ou não o elemento
```

```
            if (achou)
                System.out.println("Elemento "+ elem +" encontrado no vetor");
            else
                System.out.println("Elemento "+ elem +" NÃO encontrado no vetor");

        }
}
```

Código-fonte 48 – Rotinas utilizando vetor no Java
Fonte: Elaborado pelo autor (2022)

9 MATRIZES

Vistas as variáveis indexadas do tipo vetor, agora falaremos das variáveis bidimensionais do tipo matriz.

Antes de continuarmos falando sobre matriz, vamos ver uma ilustração comparativa entre variável comum, vetor e matriz:

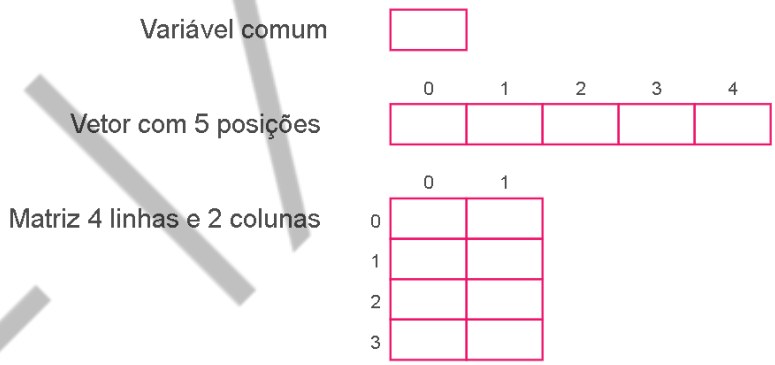

Figura 11 – Comparativo entre variável comum, vetor e matriz
Fonte: Elaborada pelo autor (2022)

Enquanto um vetor é uma variável unidimensional (1, N), a matriz é uma variável bidimensional (N, N), parecida com uma tabela. No vetor, tratávamos o seu índice apenas pela coluna, porque a linha é sempre 1:

```
vetor[2]
```

Agora na matriz, temos que referenciar também a linha. Sempre seguimos a ordem [linha] [coluna]:

```
Matriz[1][2]
```

Sendo assim, a manipulação da matriz é similar:

```
// Atribui o valor 99 na linha com o índice 2 e coluna 1
m[2][1] = 99
// permite ao usuário digitar o elemento com índices 1 e 2
Leia m[1][2]
// Exibe o primeiro elemento da matriz
Escreva "Primeira posição do vetor", m[0][0]
// Efetua cálculos e atribuições
m[2][2] = m[1][1] * m[3][3]
// Verifica se um elemento é positivo
Se m[3][1] > 0 então
    Escreva "É positivo"
Senão
    Escreva "Não é positivo"
Fim_se
```

Código-fonte 49 – Exemplos de manipulação de matriz
Fonte: Elaborado pelo autor (2022)

Caso seja colocado dentro dos colchetes um índice que extrapole o limite da matriz, ocorrerá um erro.

10 DE-PARA: MATRIZ

Vamos fazer rotinas similares às últimas feitas com vetor, mas agora usando matriz. Seguem os enunciados:

Em uma matriz de inteiros com três linhas e quatro colunas, fazer as seguintes rotinas:

1 – Preencher a matriz.

2 – Exibir o conteúdo da matriz.

3 – Somar os elementos da matriz.

4 – Buscar um elemento n.

Execução das rotinas:

```
PREENCHENDO A MATRIZ:
Matriz[0][0]=9
```

```
Matriz[0][1]=6
Matriz[0][2]=4
Matriz[0][3]=7
Matriz[1][0]=8
Matriz[1][1]=9
Matriz[1][2]=6
Matriz[1][3]=4
Matriz[2][0]=9
Matriz[2][1]=3
Matriz[2][2]=8
Matriz[2][3]=2

EXIBINDO A MATRIZ:
[0][0] = 9 [0][1] = 6 [0][2] = 4 [0][3] = 7
[1][0] = 8 [1][1] = 9 [1][2] = 6 [1][3] = 4
[2][0] = 9 [2][1] = 3 [2][2] = 8 [2][3] = 2



Somatória da Matriz = 75
Digite o elemento: 45
Elemento 45 NÃO encontrado na matriz
```

Quadro 7 – Execução das rotinas com matriz
Fonte: Elaborado pelo autor (2022)

Momento Pseudocódigo

```
Programa manipulando_matriz
Var
    Matriz[3][4]: inteiro
    l, c, soma: inteiro
    achou: boolean
Início
    // 1 - PREENCHER A MATRIZ
    Para l de 1 até 3 inc 1 faça
        Para c de 1 até 4 inc 1 faça
            Escreva "Matriz[",l,"][",c,"]= "
            Leia matriz[l][c]
        Fim_para
    Fim_para

    // 2 - EXIBIR A MATRIZ
    Para l de 1 até 3 inc 1 faça
        Para c de 1 até 4 inc 1 faça
            Escreva "[",l,"][",c,"] =", matriz[l][c]
        Fim_para
    Fim_para


    // 3 - SOMAR OS ELEMENTOS DA MATRIZ
```

```
    soma = 0
    Para l de 1 até 3 inc 1 faça
        Para c de 1 até 4 inc 1 faça
            soma = soma + matriz[l][c]
        Fim_para
    Fim_para

    Escreva "Soma = ", soma

    // 4 - BUSCAR UM ELEMENTO NO VETOR
    achou = falso
    Escreva "Digite o elemento:”
    Leia elem
    Para l de 1 até 3 inc 1 faça
        Para c de 1 até 4 inc 1 faça
            Se matriz[l][c] = elem então
                achou = verdade
                l = 3
            Fim_se
        Fim_Para
    Fim_para
    Se achou então
        Escreva "Elemento ”,elem,” encontrado na matriz”
    senão
        Escreva "Elemento ”,elem,” NÃO  foi encontrado na
matriz"
    Fim_se
Fim
```

Código-fonte 50 – Rotinas utilizando matriz no Pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
matriz = [[0,0,0,0],[0,0,0,0],[0,0,0,0]]
# 1 - PREENCHER A MATRIZ
for l in range(3):
    for c in range(4):
        matriz[l][c] = int(input(f"Matriz[{l}][{c}]= "))

# 2 - EXIBIR A MATRIZ
for l in range(3):
    for c in range(4):
        print(f"[{l}][{c}] = {matriz[l][c]}\t")
    print()

# 3 - SOMAR OS ELEMENTOS DA MATRIZ
soma = 0
for l in range(3):
```

```
    for c in range(4):
        soma += matriz[l][c]

print("Soma = ", soma)

# 4 - BUSCAR UM ELEMENTO NO VETOR
achou = False
elem = int(input("Digite o elemento:"))
for l in range(3):
    for c in range(4):
        if matriz[l][c] == elem:
            achou = True
            break

if (achou):
    print(f"Elemento {elem} encontrado na matriz")
else:
    print(f"Elemento {elem} NÃO  foi encontrado na matriz")
```

Código-fonte 51 – Rotinas utilizando matriz no Python
Fonte: Elaborado pelo autor (2022)

Momento Java

```
import java.util.Scanner;
public class First {

    public static void main(String[] args)
    {
        // Estancia o objeto Teclado para ler variáveis
        Scanner teclado = new Scanner(System.in);

        int [][] matriz = new int[3][4];

        // 1 - PREENCHER A MATRIZ
        System.out.println("\nPREENCHENDO A MATRIZ:");
        for(int l = 0; l < 3; l++){
            for(int c = 0; c < 4; c++){
                System.out.print("Matriz["+ l + "]["+ c + "]=");
                matriz[l][c] = teclado.nextInt();
            }
        }
```

```
        // 2 - EXIBIR A MATRIZ
        System.out.println("\nEXIBINDO A MATRIZ:");
        for(int l = 0; l < 3; l++){
            for(int c = 0; c < 4; c++){
                System.out.print("["+ l + "]["+ c + "] = " + matriz[l][c] +"\t");
            }
            System.out.println();
        }

        // 3 - SOMAR OS ELEMENTOS DA MATRIZ
        int soma = 0;
        for(int l = 0; l < 3; l++){
            for(int c = 0; c < 4; c++){
                soma += matriz[l][c];
            }
            System.out.println();
        }
        System.out.println("Somatória da Matriz = " + soma);

        // 4 - BUSCAR UM ELEMENTO NO VETOR
        boolean achou = false;
        int elem;

        // Digitação do elemento que será procurado
        System.out.print("Digite o elemento: ");
        elem = teclado.nextInt();

        for(int l = 0; l < 3; l++){
            for(int c = 0; c < 4; c++){
                // Caso encontre o elemento, interrompe a busca
                if (matriz[l][c] == elem)
                {
                    achou = true;
                    break;
                }
            }
        }
        // Analisa se encontrou ou não o elemento
        if (achou)
            System.out.println("Elemento "+ elem +" encontrado na matriz");
        else
            System.out.println("Elemento "+ elem +" NÃO encontrado na matriz");
```

```
    }
}
```

Código-fonte 52 – Rotinas utilizando matriz no Java
Fonte: Elaborado pelo autor (2022)

Em todos os exemplos que utilizamos com vetor e matriz, consideramos os elementos do tipo inteiro, mas isso não é regra. Poderia ser real, texto ou lógico. A ideia é a mesma: os índices são inteiros a partir do zero independentemente do tipo que o vetor armazena. Uma particularidade no Python é que, no mesmo vetor, ele pode ter elementos de tipos diferentes, enquanto em Java e Pseudocódigo os tipos dos elementos devem ser definidos.

Comparando as linguagens de programação exemplificadas acima, pudemos perceber como cada uma trata de forma peculiar a matriz (e o vetor). O que importa é que todas respeitam a essência do conceito: armazenar várias informações dentro da mesma variável, seja em forma de lista (vetor) ou tabela (matriz).

Com esses exemplos concluímos o capítulo 5 em que vimos subalgoritmos (função e procedimento) e variáveis indexadas (vetor e matriz) sendo aplicados em Pseudocódigo, Python e Java.

GLOSSÁRIO

Subalgoritmo	Subconjuntos do algoritmo.
Procedimento	Representação do subalgoritmo que não retorna valor ao programa chamador.
Função	Representação do subalgoritmo que retorna um valor ao programa chamador.
Parâmetro	Mecanismo de transporte de informações entre algoritmo e subalgoritmo.
Variável indexada	Classificação de variáveis (vetor e matriz) que utilizam o índice para referenciar um elemento.
Vetor	Variável indexada no formato de lista (1, N).
Matriz	Variável indexada no formato de tabela (N, N).