



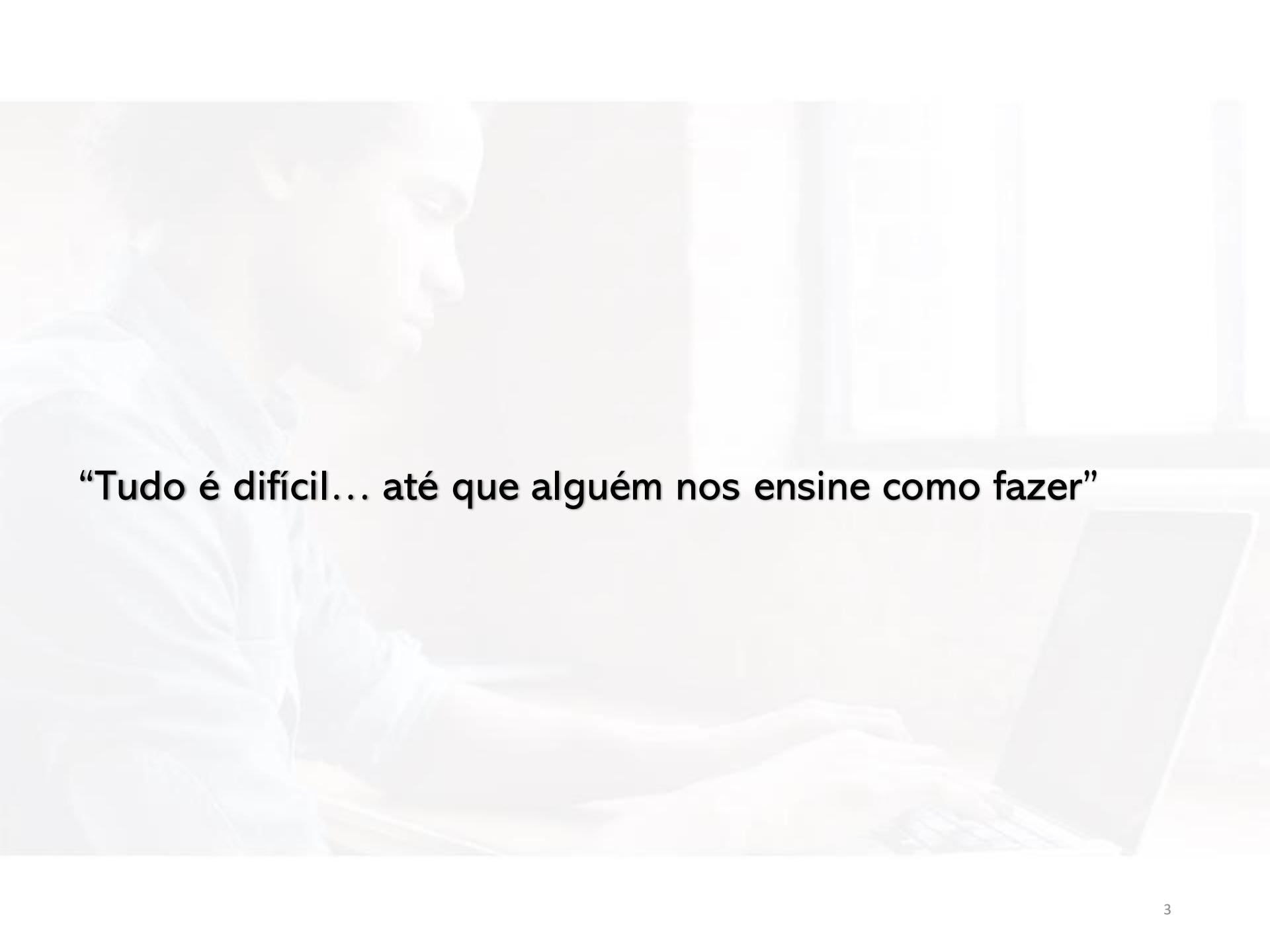
# Curso de Programação Web

**Secção 2: Programação web**  
**Módulo 3: Lógica de programação**

Osvaldo Livondeni

# Conteúdo

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Tabelas de uma dimensão (vectores)
7. Algoritmos básicos para lidar com tabelas de uma dimensão (vectores)
8. Tabelas de duas dimensões (matrizes)
9. Algoritmos básicos para lidar com tabelas de duas dimensões (matrizes)
10. Funções e procedimentos



**“Tudo é difícil... até que alguém nos ensine como fazer”**

# 1. Introdução

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

- **Análise;**
- **Concepção gráfica (template);**
- **Integração web (HTML, CSS e JavaScript);**
- **Programação lógica e criação da base de dados (PHP e MySQL);**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Para aprender a programar a gente tem de saber Lógica de Programação ou Algoritmo**

**Definição:**

- **Sequência de instruções bem definidas com o fim de solucionar um problema.**
- **Um algoritmo representa os passos necessários para realizar uma tarefa.**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Um exemplo comum de algoritmo seria uma receita de culinária onde de antemão estão definidos os ingredientes e os passos necessários para realizar um prato:**

- 1. Introdução
- 2. Instruções executáveis básicas
- 3. Operadores
- 4. Instruções condicionais
- 5. Instruções repetitivas
- 6. Vectores
- 7. Algoritmos básicos para lidar com vectores
- 8. Matrizes
- 9. Algoritmos básicos para lidar com matrizes
- 10. Funções e procedimentos

## Receita de Bolo de banana

### Ingredientes

7 bananas

1 colher (sopa) de fermento

2 chávenas (chá) de farinha de trigo ...

### Modo de preparo

1. Coloque no liquidificador os ovos, o óleo, o açúcar e, por último, o leite. Bata bem todos os ingredientes, até obter uma mistura homogênea.

2. Acrescente, aos poucos, a farinha de trigo no liquidificador ligado. Por fim, adicione o fermento. Reduza a velocidade do liquidificador e bata apenas para que a massa possa incorporar o fermento...

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Cabeçalho:** Nesta parte especificamos o nome que do algoritmo (Sem espaço nem caracter especial). Se comparada a uma receita seria o nome da receita

**Exemplo:**

Receita de Bolo de banana ...

Algoritmo Multiplicacao\_de\_dois\_numeros\_inteiros

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Parte declarativa

- Nesta parte declaramos todas as variáveis necessárias para a execução do algoritmo. Se comparada a uma receita seriam os ingredientes da receita.

## Exemplo:

1Kg de farinha, 2 ovos, 2 bananas ...

Numero1, Numero2: Inteiro

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Bloco de instruções executáveis

- Nesta parte escrevemos os passos necessários para solucionar o problema.
- É delimitado pelas palavras **Inicio** e **Fim**. Se comparada a uma receita seriam os passos necessários para realizar o prato.

## Exemplo:

Colocar o açúcar e a manteiga numa tigela e misturar bem.

Aumentar os ovos, voltar a misturar

...

Numero1 <- 8;

Numero2 <- 4;

Resultado <- Numero1 x Numero2;

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

**Algoritmo: Nome\_do\_algoritmo**

**Variáveis: <Declaração de variáveis>**

**Inicio**

**<Instruções executáveis>**

**Fim**

- 1. **Introdução**
- 2. **InSTRUções executáveis básicas**
- 3. **Operadores**
- 4. **InSTRUções condicionais**
- 5. **InSTRUções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

- 1. **Dizer as 3 partes que constituem um algoritmo.**

## 2. Instruções executáveis básicas

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Lugar na memória do computador que contem o valor utilizado durante a execução do programa.
- Um recipiente que vai albergar um valor.
- Se comparado a uma receita de culinária seria a tigela que contem o açúcar, ou o copo que contem o óleo, ou a jarra, que contem a água, ou a colher que contem a canela.

1. **Introdução**
2. **InSTRUçõEs EXECUTÁVEIS BÁSICAS**
3. **OPERADORES**
4. **INSTRUçÕES CONDICIONAIS**
5. **INSTRUçÕES REPETITIVAS**
6. **VECTORES**
7. **ALGORITMOS BÁSICOS PARA LIDAR COM VECTORES**
8. **MATRIZES**
9. **ALGORITMOS BÁSICOS PARA LIDAR COM MATRIZES**
10. **FUNçÕES E PROCEDIMENTOS**

**Se comparado com a receita de culinária, é o tamanho de uma variável:**

- **Inteiro:** São os valores numéricos inteiros.  
Ex: 5, -1, 999;
- **Decimal:** Números decimais.  
Ex: 3.4, -1.5, 5, 100
- **Boleano:** Composto por dois valores apenas - Verdadeiro ou Falso;
- **Carácter:** Para simplificar... São os valores alfabéticos e alfanuméricos  
Ex: Joao, Ana2)

**Boleano < Inteiro < Decimal < Caracter**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Nome da variável dado pelo programador.**
- **Se comparado a uma receita de culinária seriam os nomes dos recipientes: Colher de chá, colher de sopa etc.**

### **Regra:**

- **Deve começar com uma letra;**
- **Não pode conter espaço;**
- **Os únicos caracteres especiais permitidos são “-” e “\_”.**

### **Exemplos de identificadores correctos:**

- **soma, produto, Num1, Val\_Inteiro, Numero-Primo, A, a.**

### **Exemplos de identificadores incorrectos:**

- **\_soma, .produto, 1Num, Val Inteiro, Numero Primo, \*A, -a.**

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Notas ou explicações não executáveis que inserimos no algoritmo/programa.**

### **Sintaxe:**

- **// Comentário em uma linha**
- **/\* Comentário  
em várias  
linhas \*/**

1. **Introdução**
2. **InSTRUçõEs EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUçõEs CONDICIONAIS**
5. **InSTRUçõEs REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

- Na declaração de uma variável dizemos ao processador as variáveis que vamos utilizar ao longo do programa para que ele reserve os espaços necessários na memória.

### Sintaxe:

- Nome da variável: tipo
- Var1, Var2, ..., VarN: tipo

### Exemplo:

- A: caracter
- Numero1, Numero2: inteiro

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Na afectação preenchemos o lugar reservado com o respectivo valor.
- Se comparada a receita de culinária seria o acto de preencher a tigela com açúcar, ou copo com água etc.

## Sintaxe

- **Variável  $\leftarrow$  Valor (Lê-se: Variável recebe Valor)**

## Exemplo:

- **C  $\leftarrow$  5**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Usada para ler obter dados do utilizador através do teclado.

### Sintaxe

- **ler(variável)**

### Exemplo:

- **ler(A)**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Usada para mostrar algo ao utilizador através do ecrã.

### Sintaxe

- escrever("expressão")
- escrever(variavel)
- escrever("expressão", variavel)

### Exemplo

- escrever("Bem-vindo à aplicação GGPEN")
- escrever(var1)
- escrever("resultado", var1)

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Algoritmo Multiplicacao

Variáveis: A, B, produto: inteiro

### Início

escrever ("Bem-vindo a aplicação Multiplicação")

escrever ("Digitar o primeiro número")

ler (A)

escrever ("Digitar o segundo número")

ler (B)

produto  $\leftarrow$  A x B

escrever ("O resultado é: ", produto)

### Fim

### 3. Operadores

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- São expressões colocadas entre variáveis com o fim de calcular e obter um resultado.

### Exemplo:

- $A + B$
- $X < Y$
- $C \in Z$

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Os operadores são basicamente de três tipos:**

- **Aritméticos (+, -, x, /)**
- **Comparação ou relacionais (>, <, =, ≠, ≥, ≤)**
- **Lógicos (E, OU, Não)**

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

## Algoritmo: OperadoresAritmeticos

### Variáveis:

```
// Declaração de variáveis  
altura, largura: Inteiro  
X, Y, A, B, soma, divisao, subtracao, produto : Decimal
```

### Início

```
// Afectação de variáveis  
altura ← 100  
largura ← 40  
X ← 2.3  
Y ← 3.4  
A ← X  
B ← Y
```

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

```
subtracao ← altura – largura
soma ← A + B
produto ← A x B
divisao ← X/Y
escrever("A subtração é", subtracao)
escrever("A soma é", soma)
escrever("O produto é", produto)
```

Fim

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Nota importante:** O resultado de uma operação de comparação é de tipo booleano – Verdade ou Falso.

**Algoritmo: Comparacao**

**Variáveis:**

```
// Declaração de variáveis
X, Y: Inteiro
A, B, C, D, E, F: Booleano
```

**Inicio**

```
// Afectação de variáveis
X ← 10
Y ← 20
A ← X < Y // A recebe Verdade
B ← X > Y // B recebe Falso
C ← X = Y // C recebe Verdade
D ← X ≠ Y // D recebe Falso
E ← X ≤ Y // E recebe Verdade
F ← (X ≥ Y) // F recebe Falso
```

**Fim**

- 1. Introdução
- 2. Instruções executáveis básicas
- 3. Operadores
- 4. Instruções condicionais
- 5. Instruções repetitivas
- 6. Vectores
- 7. Algoritmos básicos para lidar com vectores
- 8. Matrizes
- 9. Algoritmos básicos para lidar com matrizes
- 10. Funções e procedimentos

- Os operadores lógicos mais usados são E (AND), OU (OR), NÃO (NOT).

## Nota importante:

- São utilizados para operar variáveis do tipo booleano (Verdade ou Falso).
- O resultado de uma operação lógica é de tipo Booleano.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

## Operador lógico E (*AND*)

O resultado será *Verdade* quando *A* e *B* forem *Verdade*.

Tabela de verdade do operador lógico E

A	B	A E B
Falso	Falso	Falso
Falso	Verdade	Falso
Verdade	Falso	Falso
Verdade	Verdade	Verdade

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

## Operador lógico OU (*OR*)

O resultado será *Verdade* quando  $A \text{ OU } B$  for *Verdade*.

Tabela de verdade do operador lógico OU

A	B	A OU B
Falso	Falso	Falso
Falso	Verdade	Verdade
Verdade	Falso	Verdade
Verdade	Verdade	Verdade

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

## Operador lógico NÃO (*NOT*)

O resultado é o oposto do operando

### Tabela de verdade do operador lógico NÃO

A	NÃO (A)
Falso	Verdade
Verdade	Falso

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

## Algoritmo: Operadores\_logicos

Variáveis:

X, Y: Inteiro

A, B, C, D: Booleano

Inicio

// Afectação de variáveis

X ← 10

Y ← 20

A ← (X < Y) E (X > Y); /\* A recebe Falso porque temos (Verdade E Falso) \*/

C ← (X ≤ Y) OU (X = Y); /\* C recebe Verdade porque temos (Verdade OU Falso) \*/

B ← NÃO (X > Y) /\* B recebe Verdade porque temos NÃO (Falso) \*/

Fim

## 4. Instruções Condicionais

Se a temperatura do fogo estiver a 100 graus, parar de aquecer...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Ao executarmos uma receita de culinária, poderemos ver que certos passos só são executados em certas circunstâncias.
- Esses passos só são executados caso algo de especial aconteça.

Exemplo:

- Abrir a tampa da panela se a água transbordar por causa da temperatura.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Em programação também, as vezes nós queremos que certa parte do programa seja executada apenas em determinada condição.
- Nestes casos usamos as instruções condicionais.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Se comparada a receita do bolo, seria:**

- Se a temperatura do forno chegar a 50°C, então reduzir para 30°

**Em algoritmo, seria mais ou menos assim:**

- Se  $T = 50$  Então  $T \leftarrow 30$

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Existem basicamente 2 tipos de instruções condicionais:

- Tipo 1: Se ... Então ...
- Tipo 2: Se ... Então ... Se não ...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

É o tipo mais simples. Igual o do exemplo, ou seja:

- Se a temperatura do forno chegar a 50°C, então reduzir para 30°

Em algoritmo:

- Se  $T = 50$  Então  $T \leftarrow 30$  Fim Se

Sintaxe:

- Se <condição> Então <instrução> Fim Se

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- É o tipo um pouco mais elaborado do tipo 1.
- Neste tipo, especificamos a condição em que a instrução será realizada, e também especificamos uma segunda instrução a ser realizada caso a condição não for verificada.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Exemplo:

- Se a temperatura do forno chegar a 50°C, então reduzir para 30°.
- Se a temperatura do forno não chegar a 50°C, então subir a temperatura para 40°C

### Em algoritmo:

Se  $T = 50$  Então  $T \leftarrow 30$

Se não  $T = 40$

Fim Se

### Sintaxe:

Se <condição> Então <instrução 1>

Se não <instrução 2>

Fim Se

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Neste exemplo vamos escrever o algoritmo de uma aplicação que realiza a operação de divisão, com os seguintes passos:

- O programa pede para o utilizador introduzir o primeiro número;
- O programa lê o primeiro número que o utilizador digita;
- O programa pede para o utilizador introduzir o segundo número;
- O programa lê o segundo número que o utilizador digita;
- Se o segundo número introduzido pelo utilizador for igual a zero, o programa mostra a mensagem que não é possível dividir um número por zero.
- Caso contrário, o programa efectua a divisão dos dois números e mostra o resultado;

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Algoritmo: Divisao

Variáveis:

x, y: Decimal;  
divisao: Decimal;

Inicio

escrever("Bem-vindo à aplicação Divisão")  
escrever("Digite o primeiro número")  
ler(x)  
escrever("Digite o segundo número")  
ler(y)

// Verificar o valor de y

Se y=0 Então

escrever("Não é possível dividir um número por zero")

Se não

// Operação de divisão

divisao ← x/y

// Mostrar o resultado

escrever("O resultado é", divisao)

Fim Se

Fim

## 5. Instruções repetitivas

Mexer a massa **n vezes** até ficar homogênea ...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Imagina que vai escrever o algoritmo de uma aplicação que mostra a sequência de números inteiros de 1 à 10 com o que aprendeu até aqui ... – Fácil (mais 10 linhas de código)
- Ou escrever o algoritmo de uma aplicação que mostra a sequência de números inteiros de 1 à 100 ... com o que aprendeu até aqui ... – Começar a dificuldade (mais 100 linhas de código)
- Ou escrever o algoritmo de uma aplicação que mostra a sequência de números inteiros de 1 à 1000 com o que aprendeu até aqui ... – Muito cansativo e repetitivo (mais 1000 linhas de código)...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Como podemos ver, quanto maior for o número de repetições, mais extenso será o nosso algoritmo.
- Felizmente existem as Instruções Repetitivas para que o processador execute essas repetições automaticamente.
- Sendo assim, a utilidade das instruções repetitivas é automatizar tarefas que se repetem.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Tomemos o seguinte exemplo da nossa receita de bolo:
  - Ao fazer o bolo, depois de meter os ingredientes, temos de misturar até a massa ficar homogênea.
  - Quem já fez ou viu a fazer um bolo sabe o quanto cansativo é misturar a massa do bolo à mão. A gente mistura, mistura e mistura mais...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Agora imagina um dispositivo que realiza as misturas automaticamente.
- Um dispositivo tipo o misturador que com apenas uma apertada no botão ele mistura, mistura e mistura, até a massa ficar homogênea.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Em programação o nosso dispositivo de misturas automáticas são as instruções repetitivas.
- Elas são muito importantes porque nos permitem automatizar tarefas que se repetem.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Existem basicamente três tipos de instruções repetitivas:**

- **Tipo 1: De ... À ... Fazer**
- **Tipo 2: Enquanto ... Fazer**
- **Tipo 3: Fazer ... Até que**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Esta instrução repetitiva é utilizada quando sabemos de antemão o número de repetições. Como, escrever de 1 à 1000.**

### **Sintaxe:**

- de <variável> ← <valor de início> à <valor de término> fazer
  - <instruções>
  - fim de

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### **Exemplo:**

- **Escrever de 1 à 1000.**

### **Em algoritmo:**

- de numero ← 1 à 1000 fazer  
                  escrever ( numero )  
                  fim de

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Esta instrução é utilizada quando queremos que uma instrução ou bloco de instruções seja executado repetidamente enquanto uma condição for Verdade.
- Ou seja, sabemos o que queremos, mas não sabemos quantas vezes serão necessárias repetir o processo até atingir isso.
- Como misturar a massa até ficar homogênea (Quantas vezes teremos de misturar? R: Não sabemos! )

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### **Sintaxe:**

- enquanto (condição) fazer
  - <instruções>
  - fim enquanto

### **Exemplo:**

- enquanto (Massa ≠ Homogenea) fazer
  - Misturar ()
  - fim enquanto

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### OBS:

- A Condição deve ser uma variável de tipo booleano ou uma expressão que tem como resultado um booleano (verdade ou falso).

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Esta instrução é semelhante a passada, Enquanto ... Fazer ...
- É utilizada quando queremos que uma instrução ou bloco de instruções seja executado repetidamente até que uma condição seja Verdade.
- Ou seja, sabemos o que queremos, mas não sabemos quantas vezes serão necessárias repetir o processo até atingir isso.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Sintaxe:

- **fazer**  
    <instruções>  
    até que (condição)

### Exemplo:

- Misturar a massa até ficar homogênea (Quantas vezes teremos de misturar? R: Não sabemos! )

### Em algoritmo:

- **fazer**  
    Misturar ()  
    até que (Massa = Homogênea)

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### OBS:

- A grande particularidade deste tipo de instrução repetitiva está no facto de ela executar as *instruções* antes de verificar a *condição*;
- Então, as *instruções* serão executadas pelo menos uma vez mesmo que a *condição* não permita.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- **Neste exemplo vamos escrever o algoritmo de uma aplicação que realiza a contagem de 1 à um número desejado pelo utilizador, com os seguintes passos:**
  - O programa pede ao utilizador introduzir o número limite de contagem;
  - O programa lê o número que o utilizador digita;
  - O programa efectua a contagem até o número limite, mostrando no ecrã a contagem.

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Algoritmo: Contagem-Crescente

Variáveis:

```
i: inteiro;  
limite: inteiro;
```

Início

```
    escrever("Bem-vindo à aplicação Contagem Crescente")  
    escrever("Digite o numero limite para a  
    contagem")
```

```
    ler(limite)
```

```
    /* Executar a contagem usando a instrução  
    repetitiva do tipo 1 */
```

```
    de i ← 1 à limite fazer
```

```
        escrever(i)
```

```
    fim de
```

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
/* Executar a contagem usando a instrução  
repetitiva do tipo 2 */
```

```
i←1  
enquanto (i ≠ limite) fazer  
    escrever(i)  
    i++ //é chamado "incremento"  
fim enquanto
```

```
/* Executar a contagem usando a instrução  
repetitiva do tipo 3 */
```

```
i←1  
fazer  
    escrever(i)  
    i++ //é chamado "incremento"  
até que (i = limite)  
fim
```

## 6. Tabelas de uma dimensão (vectores)

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Em programação, uma tabela é simplesmente uma estrutura de dados usada para armazenar de forma organizada várias informações relacionadas umas com as outras.

Por exemplo:

- Os números das camisolas dos 11 jogadores de futebol de uma equipa,
- Os nomes dos estudantes de uma turma,
- Os nomes dos habitantes de um bairro ...

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Voltemos a nossa comparação com a receita de culinária.**
- **Lembram-se quando definimos as Variáveis?**  
Pois, dissemos que uma Variável era um recipiente que albergava um valor.
- **Lembram-se com que comparamos as variáveis na receita de culinária?** Pois, dissemos que eram como a tijela (que contem o açucar), o copo (que contem o óleo), a jarra (que contem a agua) ...

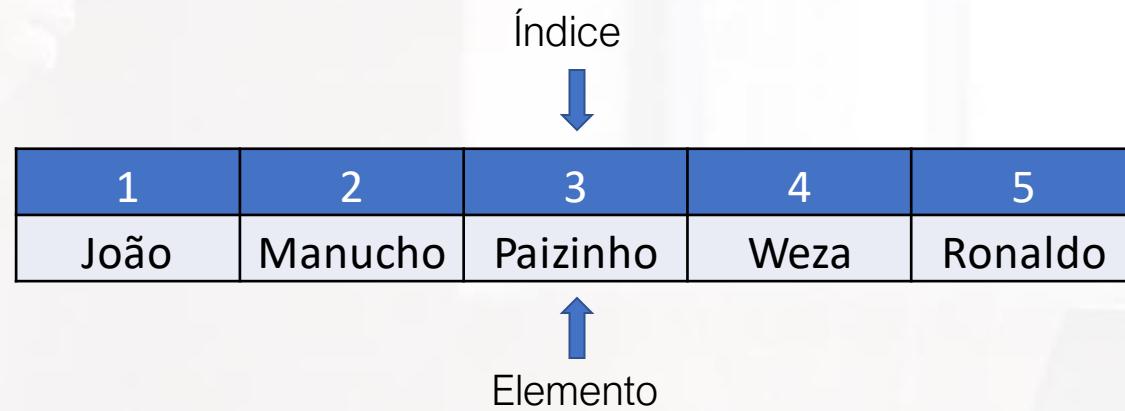
1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Por sua vez, a tabela seria um tipo de recipiente com vários compartimentos, que nem aquelas bandejas grandes usadas nos refeitórios onde colocamos o arroz num compartimento, o peixe num outro, o molho também num outro etc.



1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Representação gráfica de uma tabela:



- **Índice:** Serve para indicar a posição do *elemento* na tabela
- **Elemento:** Compartimento na tabela para armazenar um valor.

1. **Introdução**
2. **InSTRUçõEs  
executáveis  
básicas**
3. **Operadores**
4. **InSTRUçõEs  
condicionais**
5. **InSTRUçõEs  
repetitivas**
6. **Vectores**
7. **Algoritmos  
básicos para lidar  
com vectores**
8. **Matrizes**
9. **Algoritmos  
básicos para lidar  
com matrizes**
10. **Funções e  
procedimentos**

- Declaramos uma tabela escrevendo o nome da tabela, dois pontos (:), seguido da palavra tabela de;
- A seguir especificamos o tipo de variável que ela vai albergar;
- Por último especificamos a quantidade de compartimentos entre parênteses rectos.

### Sintaxe:

- `Nome_da_tabela : tabela de <Tipo_de_variável> [1..<num de elementos>]`

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

### Exemplos:

- tab1: tabela de Inteiro [1..10]; // Declaração de uma tabela que armazenará 10 valores inteiros
- tab2: tabela de Real [1..15]; // Declaração de uma tabela que armazenará 15 valores reais
- tab3: tabela de Carácter [1..100]; // Declaração de uma tabela que armazenará 100 caracteres.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Quando declaramos uma tabela:**
  - `T : tabela de Inteiro [1..10];`
- **Estamos na verdade a definir uma tabela de 10 compartimentos indexados de 1 a 10.**

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Para acedermos a um compartimento da tabela especificamos o nome da tabela seguido do índice do compartimento entre parênteses rectos.

Exemplos:

1	2	3	4	5	6	7	8	9	10
1	100	8	9	856	7	0	18	99	36

↑  
T[1]

↑  
T[3]

↑  
T[10]

- Aceder o primeiro elemento de T:
  - $T[1]$
- Aceder o terceiro elemento de T:
  - $T[3]$
- Aceder o décimo elemento de T:
  - $T[10]$

1. **Introdução**
2. **InSTRUÇÕES  
EXECUTÁVEIS  
BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES  
CONDICIONAIS**
5. **InSTRUÇÕES  
REPETITIVAS**
6. **Vectores**
7. **Algoritmos  
bÁSICOS PARA LIDAR  
COM VECTORES**
8. **Matrizes**
9. **Algoritmos  
bÁSICOS PARA LIDAR  
COM MATRIZES**
10. **Funções e  
procedimentos**

- **Para afectarmos uma tabela, fazemo-lo de três formas diferentes:**
  - Na declaração da tabela;
  - Especificando o comportamento que vai receber o valor;
  - Usando uma instrução repetitiva.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Na declaração da tabela:**

- `numeros_barcelona: tabela de inteiro [1..5] ← {4, 1, 2, 7, 6}`

**Teremos:**

1	2	3	4	5
4	1	2	7	6

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Especificando o compartimento que vai receber o valor:**

- **Dada a seguinte tabela:**
  - nomes\_barcelona: tabela de carácter [1..3]
- **Para afectarmos o valor “Messi” no 1º compartimento fazemos o seguinte:**
  - nomes\_barcelona[1] ← “Messi”
- **Para afectarmos o valor “Iniesta” no 2º compartimento fazemos o seguinte:**
  - nomes\_barcelona[2] ← “Iniesta”
- **Para afectarmos o valor “Piqué” no 3º compartimento fazemos o seguinte:**
  - nomes\_barcelona[3] ← “Piqué”

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Teremos então:

1	2	3
Messi	Iniesta	Piqué

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Usando a instrução repetitiva:

```
de i ← 1 à 3 fazer
    tabela_unidade [i] ← 1
fim de
```

- Teremos:

1	2	3
1	1	1

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Mostrar o conteúdo de alguns compartimentos de uma tabela:**

- Já é sabido que para mostrar algo no ecrã usa-se a instrução de saída/output de dados `escrever()`;
- E para aceder um compartimento da tabela usa-se o nome da tabela seguido do índice do mesmo colocado entre parênteses recto;
- Então para mostrar o valor/conteúdo de um compartimento da tabela usa-se uma combinação destas duas instruções.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### Exemplo:

- **Dada a tabela**

- `nomes_barcelona: tabela de caracter [1..3]  
  ← {"Messi", "Iniesta", "Piqué"}`

- **Vamos mostrar no ecrã o seu 2º elemento:**

- `escrever(nomes_barcelona[2]) /* Vai  
mostrar o valor Iniesta no ecrã */`

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Mostrar o conteúdo de toda a tabela

- Visto que os compartimentos de uma tabela estão indexados de forma contínua: 1, 2, 3, ..., N
- Para mostrar **todos** os elementos de uma tabela temos de usar uma estrutura repetitiva.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### Exemplo:

- **Dada a tabela**
  - `nomes_barcelona`: tabela de caracter [1..3]  
   $\leftarrow \{ \text{"Messi"}, \text{"Iniesta"}, \text{"Piqué"} \}$
- **Para mostrar todos os elementos de T fazemos:**

```
de i ← 1 à 3 fazer
    escrever (nomes_barcelona [i])
fim de
```

# 7. Algoritmos básicos para lidar com tabelas de uma dimensão (vectores)

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Os algoritmos básicos para lidar com tabelas são a base para assimilar o raciocínio lógico de algoritmos complexos.
- Tudo que quero que vocês retenham são os CONCEITOS.
- Não tentem decorar os algoritmos, porque vão esquecer daqui a algum tempo – Eu só me estou a recordar deles agora que estou a vos ensinar...

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

**Os algoritmos básicos para lidar com tabelas são:**

- Intercambiar dois elementos de uma tabela;
- Procurar um elemento na tabela
- Procurar o mínimo
- Procurar o máximo
- Procurar a posição do mínimo
- Procurar a posição do máximo
- Algoritmo de ordenação – Sorting algorit

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Imaginemos que temos um copo com água e um outro com azeite.**
- **Como faria se lhe pedissem para intercambiar o conteúdo dos dois copos, ou seja, meter a água no copo de azeite, e o azeite no copo de água?**

- 1. Introdução
- 2. Instruções executáveis básicas
- 3. Operadores
- 4. Instruções condicionais
- 5. Instruções repetitivas
- 6. Vectores
- 7. Algoritmos básicos para lidar com vectores
- 8. Matrizes
- 9. Algoritmos básicos para lidar com matrizes
- 10. Funções e procedimentos

- Se quisermos intercambiar o conteúdo dos dois copos precisaremos de um copo adicional e agiríamos da seguinte forma:
  1. Colocaríamos a água no recipiente adicional;
  2. Colocaríamos em seguida o azeite no copo onde estava a água;
  3. Para terminar colocaríamos a água do recipiente adicional para o recipiente onde estava o azeite.

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- O mesmo fazemos em programação para intercambiar valores de duas variáveis.
- Dadas as variáveis `variavel1←5` e `variavel2←10`.
- Para intercambiarmos os seus respectivos valores fazemos o seguinte:
  1. `temp ← variavel1` /\* Colocamos o valor de variavel1 numa variável temporária \*/
  2. `variavel1 ← variavel2`
  3. `variavel2 ← temp` /\* Colocamos em variavel2 o valor da variável temporária \*/

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Exemplo:**

- Dada a seguinte tabela `nomes_barcelona`: tabela de carácter  $[1..3] \leftarrow \{\text{"Messi"}, \text{"Iniesta"}, \text{"Piqué"}\}$

1	2	3
Messi	Iniesta	Piqué

Intercambiemos o 1º e 3 elemento:

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

1. *temp*  $\leftarrow$  *nomes\_barcelona* [1] /\* Colocamos o valor do primeiro compartimento numa variável temporária \*/
2. *nomes\_barcelona* [1]  $\leftarrow$  *nomes\_barcelona* [3] /\* Colocamos no primeiro compartimento o valor do terceiro \*/
3. *nomes\_barcelona* [3]  $\leftarrow$  *temp* /\* Colocamos no terceiro compartimento o valor da variável temporária \*/

No final teremos:

1	2	3
Piqué	Iniesta	Messi

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Para procurar um elemento numa tabela usamos uma instrução repetitiva para percorrer toda a tabela;
- Em seguida usamos uma instrução condicional para testar se um elemento da tabela é igual ao valor procurado.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Exemplo:

- Dada a seguinte tabela nomes\_barcelona: tabela de carácter [1..3]  $\leftarrow \{ \text{"Messi"}, \text{"Iniesta"}, \text{"Piqué"} \}$

1	2	3
Messi	Iniesta	Piqué

- Procuremos o elemento “Iniesta”.

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

```
de i ← 1 à 3 fazer
    se (nomes_barcelona[i] = "Iniesta") então
        escrever ("Elemento encontrado")
    fim se
fim de
```

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

- Para procurarmos o menor elemento de uma tabela, começamos por supor que o primeiro elemento é o menor da tabela;
- Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela;
- Se um dos elementos for menor, então este passa a ser o novo menor;
- Repetimos este processo até que já não haja novo menor.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Dada a seguinte tabela

- `T: tabela de inteiro[1..5] ← {100, 10, 8, 3, 856}`

1	2	3	4	5
100	10	8	3	856

- Procuremos o mínimo (menor elemento):

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
min ← T[1]
de i ← 1 à 5 fazer
    se (T[i] < min) então
        min ← T[i]
    fim se
fim de
```

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

- Para procurarmos o maior elemento de uma tabela, começamos por supor que o primeiro elemento é o maior da tabela;
- Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela;
- Se um dos elementos for maior, então este passa a ser o novo maior;
- Repetimos este processo até que já não haja novo maior.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Exemplo:

- Dada a seguinte tabela
  - T: tabela de inteiro[1..5]  $\leftarrow \{100, 10, 8, 3, 856\}$

1	2	3	4	5
100	10	8	3	856

- Procuremos o máximo (maior elemento):

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
max ← T[1]
de i ← 1 à 5 fazer
    se (T[i] > max) então
        max ← T[i]
    fim se
fim de
```

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Para procurarmos a posição do menor elemento de uma tabela, agimos da mesma forma que ao procurar o menor elemento, ou seja, começamos por supor que a primeira posição (índice) é a posição do menor elemento da tabela;
- Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o elemento da primeira posição com os restantes elementos da tabela;
- Se um dos elementos for menor, então a posição deste, passa a ser a posição do novo menor. Repetimos este processo até que já não haja novo menor.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Exemplo

- Dada a seguinte tabela
  - T: tabela de inteiro[1..5]  $\leftarrow \{100, 10, 8, 3, 856\}$

1	2	3	4	5
100	10	8	3	856

- Procuremos a posição do mínimo:

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
posMin ← 1;  
de i ← 1 à 5 fazer  
    se (T[i] < T[posMin]) então  
        posMin ← i;  
    fim se  
fim de
```

- 1. **Introdução**
- 2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
- 3. **Operadores**
- 4. **InSTRUÇÕES CONDICIONAIS**
- 5. **InSTRUÇÕES REPETITIVAS**
- 6. **Vectores**
- 7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
- 8. **Matrizes**
- 9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
- 10. **Funções e procedimentos**

- De mesmo, para procurarmos a posição do maior elemento de uma tabela, começamos por supor que a primeira posição é a posição do maior elemento da tabela.
- Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela.
- Se um dos elementos for maior, então a posição deste passa a ser a posição do novo maior. Repetimos este processo até que já não haja novo maior.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Dada a seguinte tabela

- T: tabela de inteiro[1..5]  $\leftarrow \{100, 10, 8, 3, 856\}$

1	2	3	4	5
100	10	8	3	856

- Procuremos a posição do mínimo:

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
posMax ← 1
de i ← 1 à 5 fazer
    se (T[i] > T[posMax]) então
        posMax ← i;
    fim se
fim de
```

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Ordenar uma tabela consiste em organizar os seus elementos em ordem crescente, descrente ou numa outra ordem específica.
- Existem vários tipos de algoritmo de ordenação e nós vamos aprender o mais simples dentre eles -Algoritmo de ordenação à bolha (Bubble sort algorithm).

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

O algoritmo de ordenamento à bolha tem o seguinte princípio:

- A ideia é percorrer o vector diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência. (Wikipedia)

Passo a passo:

- A gente percorre a tabela;
- Compara cada par de elementos vizinhos;
- Se o par não estiver ordenado, a gente intercambia eles de modo á ordená-los.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Tomemos a seguinte tabela não ordenada como exemplo:

14	33	27	35	10
----	----	----	----	----

- Para ordená-la em ordem crescente usando o algoritmo de ordenação à bolha, começamos com os primeiros dois elementos, comparando-os para verificar qual deles é o maior.
- Neste caso, o valor 33 é maior que 14, portanto, ele já está classificado:

14	33	27	35	10
----	----	----	----	----

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Em seguida, comparamos 33 com 27:

14	33	27	35	10
----	----	----	----	----

- Verificamos que 33 é maior que 27 e esses dois valores devem ser intercambiados:

14	33	27	35	10
----	----	----	----	----

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

A nova tabela fica assim:

14	27	33	35	10
----	----	----	----	----

Em seguida, comparamos 33 e 35. Verificamos que estão em posições já classificadas:

14	27	33	35	10
----	----	----	----	----

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Então nos movemos para os próximos dois valores, 35 e 10:

14	27	33	35	10
----	----	----	----	----

Verificamos que 35 é maior que 10. E que eles não estão ordenados:

14	27	33	35	10
----	----	----	----	----

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Intercambiaremos esses valores. E verificamos que chegamos ao final da tabela. Depois de uma iteração, a tabela ficará assim:

14	27	33	10	35
----	----	----	----	----

Depois da segunda iteração, a tabela ficará assim:

14	27	10	33	35
----	----	----	----	----

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Observe que depois de cada iteração, pelo menos um elemento “sobe para cima”:

14	10	27	33	35
----	----	----	----	----

E quando não for mais necessário intercambiar elementos, o algoritmo supõe que a tabela está completamente ordenada:

10	14	27	33	35
----	----	----	----	----

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Dada a seguinte tabela T: tabela de inteiro[1..5]  $\leftarrow \{100, 10, 8, 3, 856\}$

1	2	3	4	5
100	10	8	3	856

- Para classificá-la em ordem crescente procedemos da seguinte forma:

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
N←5 // Tamanho da tabela
de i ← 1 à N-1 fazer
    de j ← i+1 à N fazer
        se (T[i] > T[j]) então
            temp ← T[i]
            T[i] ← T[j]
            T[j] ← temp
        fim se
    fim de
fim de
```

## 8. Tabelas de duas dimensões(Matrizes)

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Lembram-se da definição de uma tabela em programação?
  - Em programação uma tabela é uma estrutura de dado usada para organizar várias informações relacionadas uma com a outra.
  - Por exemplo, os números das camisolas dos 11 jogadores de futebol de uma equipa, os nomes dos 11 jogadores da mesma equipa, etc.

- 1. Introdução
- 2. Instruções executáveis básicas
- 3. Operadores
- 4. Instruções condicionais
- 5. Instruções repetitivas
- 6. Vectores
- 7. Algoritmos básicos para lidar com vectores
- 8. Matrizes
- 9. Algoritmos básicos para lidar com matrizes
- 10. Funções e procedimentos

- Se quiséssemos armazenar numa só variável os números das camisolas dos 11 jogadores de futebol de uma equipa:
  - Seria mais fácil se essa variável fosse do tipo Tabela de Inteiro.
- Se quiséssemos armazenar numa só variável os nomes dos 11 jogadores da mesma equipa:
  - Seria mais fácil se essa variável fosse do tipo ou Tabela de Carácter.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Tabela de número de 11 jogadores de uma equipa de futebol:**

1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	----	----

- **Tabela de nomes de 11 jogadores de uma equipa de futebol:**

Marito	Bodunha	Mendonça	Paulo Silva	Paulão	Akwá	Zico	Quinzinho	Betinho	Zé Calanga	Caxaramba
--------	---------	----------	-------------	--------	------	------	-----------	---------	------------	-----------

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Imaginemos que queiramos armazenar as duas informações numa mesma variável, afim de sabermos o número da camisa e o nome do jogador correspondente.
- Para tal, existem as tabelas de duas dimensões:

Tabela de números de jogadores

1	2	14	15	9	10	10	15	7	17	11
Marito	Bodunha	Mendonça	Paulo Silva	Paulão	Akwá	Zico	Quinzinho	Betinho	Zé Calanga	Caxaramba

Tabela de nomes de jogares

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Definição:** Uma tabela de duas dimensões é uma união de duas ou mais tabelas de uma dimensão, onde:

- O número de linhas **I** é o quantidade de tabelas,
- O número de colunas **J** é a tamanho das tabelas.

J: Número de colunas – Tamanho das tabelas – Índice de colunas



	1	2	3	4	5	6	7	8	9	10	11
1	1	2	14	15	9	10	10	15	7	17	11
2	Marito	Bodunha	Mendonça	Paulo Silva	Paulão	Akwá	Zico	Quinzinho	Betinho	Zé Calanga	Caxaramba



I: Número de linhas – Quantidade de tabelas – Índice de linhas

1. **Introdução**
2. **InSTRUçõEs EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUçõEs CONDICIONAIS**
5. **InSTRUçõEs REPEtITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

- Declaramos uma tabela escrevendo o nome da tabela, dois pontos (: ), seguido da palavra tabela de.
- A seguir especificamos o tipo de variável que ela vai albergar
- Por último especificamos o número de linhas e número de colunas entre parênteses rectos, separados por uma vírgula.

### Sintaxe:

- `Nome_da_tabela : tabela de <Tipo_de_variável> [I, J]`

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### Exemplos:

- tab1: tabela de Inteiro [3,5] // Declaração de uma tabela de inteiro de 3 linhas e 5 colunas
- tab2: tabela de Real [4,4] // Declaração de uma tabela de real de 4 linhas e 4 colunas
- tab3: tabela de Carácter [6,5] // Declaração de uma tabela de carácter de 6 linhas e 5 colunas

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Quando declaramos uma tabela de duas dimensões:  
palancas\_negras: tabela de caracter [2,11];

Estamos na verdade a definir uma tabela de  $2 \times 11$  compartimentos indexados de 1,1 à 2,11:

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10	2,11

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Para acedermos um compartimento da tabela de duas dimensões:
  - Especificamos o nome da tabela
  - seguido dos índices da linha e da coluna, separados por uma vírgula, entre parênteses rectos:

### Sintaxe

<nome da tabela> [i, j]

Para acedermos o compartimento da segunda linha, quarta coluna fizemos o seguinte: `palanca_negra[2, 4]`.

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10	2,11

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

### Exemplo:

- Para acedermos o compartimento da segunda linha, quarta coluna fizemos o seguinte:

```
palancas _ negras [2 , 4]
```

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10	2,11

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Para afectarmos uma tabela de duas dimensões, fazemo-lo especificando o compartimento que vai receber o valor.

Exemplo:

```
palanca_negra [1, 6] ← "10"  
palanca_negra [2, 6] ← "Akwá"
```

Teremos:

	1	2	3	4	5	6	7	8	9	10	11
1						10					
2						Akwá					

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Uma outra maneira, consiste em usarmos as estruturas repetitivas.
- Dada T: Tabela de Inteiro [2, 11], vamos preencher-la com números de 1 à 22:

```
elemento ← 1;  
de i ← 1 à 2 fazer  
    de j ← 1 à 11 fazer  
        T[i, j] ← elemento  
        elemento ← elemento + 1  
    fim de  
fim de
```

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Teremos:

	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	5	6	7	8	9	10	11
2	12	13	14	15	16	17	18	19	20	21	22

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Mostrar o conteúdo de alguns compartimentos de uma tabela**

- Já é sabido que para mostrar algo no ecrã usa-se a instrução de saída/output de dados *escrever( )*;
- E para aceder um compartimento da tabela usa-se o nome da tabela seguido do índice do mesmo colocado entre parênteses recto;
- Então, para mostrar o valor/conteúdo de um compartimento da tabela usa-se uma combinação destas duas instruções.

**Sintaxe:**

*escrever (<nome da tabela[i,j]>)*

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

### Exemplo:

- **Dada a tabela**

palanca\_negra: tabela de caracter [2,11]

- **Vamos mostrar no ecrã o elemento da 2<sup>a</sup> linha e 6<sup>a</sup> coluna:**

escrever (palanca\_negra [2, 6])

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

### Mostrar o conteúdo de toda a tabela

- Para mostrar todos os elementos de uma tabela temos de usar uma estrutura repetitiva.

### Exemplo:

- Dada a tabela

palanca\_negra: tabela de caracter [2,11]

- Para mostrar todos os elementos da tabela fazemos:

de i ← 1 à 2 fazer

    de j ← 1 à 11 fazer

        escrever ( palanca\_negra [i, j] )

    fim de

fim de

# 9. Algoritmos básicos para lidar com tabelas de duas dimensões (Matrizes)

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Imaginemos que temos um copo com água e um outro com azeite
- Como faria se lhe pedissem para intercambiar o conteúdo dos dois copos, ou seja, meter a água no copo de azeite, e o azeite no copo de água?

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Se quisermos intercambiar o conteúdo dos dois copos precisaremos de um copo adicional e agiríamos da seguinte forma:

1. Colocaríamos a água no recipiente adicional;
2. Colocaríamos em seguida o azeite no copo onde estava a água;
3. Para terminar colocaríamos a água do recipiente adicional para o recipiente onde estava o azeite.

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Para intercambiar elementos de uma tabela, agimos então da seguinte forma:

- **Dada a seguinte tabela:**

- `nomes _ numeros _ angola`: tabela de carácter [2, 11]

1	2	14	15	9	10	10	15	7	17	11
Marito	Bodunha	Mendonça	Paulo Silva	Paulão	Akwá	Zico	Quinzinho	Betinho	Zé Calanga	Caxaramba

- **Intercambiemos o 1º e o 3º elemento da primeira linha:**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

1. 

```
temp ← nomes_numeros_angola [1,1] /* Colocamos o valor do primeiro compartimento da primeira linha numa variável temporária */
```
2. 

```
nomes_numeros_angola[1,1] ← nomes_numeros_angola[1,3]
```

/\* Colocamos no primeiro compartimento da primeira linha o valor do terceiro \*/
1. 

```
nomes_numeros_angola [1,3] ← temp /* Colocamos no terceiro compartimento o valor da variável temporária */
```

No final teremos:

14	2	1	15	9	10	10	15	7	17	11
Marito	Bodunha	Mendonça	Paulo Silva	Paulão	Akwá	Zico	Quinzinho	Betinho	Zé Calanga	Caxaramba

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Para procurar um elemento numa tabela usamos uma instrução repetitiva para percorrer toda a tabela;
- Em seguida usamos uma instrução condicional para testar se um elemento da tabela é igual ao valor procurado.

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

- **Dada a seguinte tabela:**

nomes\_numeros\_angola: tabela de carácter [2, 11]

- **Procuremos o elemento “Akwá”:**

de i ← 1 à 2 fazer

    de j ← 1 à 11 fazer

        se (nomes\_numeros\_angola[i,j] = “Akwá”) então

            escrever (“Elemento encontrado”)

        fim se

    fim de

fim de

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Para procurarmos o menor elemento de uma tabela:**
  - Começamos por supor que o primeiro elemento é o menor da tabela;
  - Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela;
  - Se um dos elementos for menor, então este passa a ser o novo menor;
  - Repetimos este processo até que já não haja novo menor.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Dada a seguinte tabela:**
  - T: tabela de inteiro[3, 4]
- **Procuremos o mínimo (menor elemento):**

```
min ← T[1,1]
de i ← 1 à 3 fazer
    de j ← 1 à 4 fazer
        se (T[i,j] < min) então
            min ← T[i,j]
        fim se
    fim de
fim de
```

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Para procurarmos o maior elemento de uma tabela:**
  - Começamos por supor que o primeiro elemento é o maior da tabela;
  - Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela;
  - Se um dos elementos for maior, então este passa a ser o novo maior;
  - Repetimos este processo até que já não haja novo maior.

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Dada a seguinte tabela:**
  - T: tabela de inteiro[3, 4]
- **Procuremos o máximo (maior elemento):**

```
max ← T[1,1]
de i ← 1 à 3 fazer
    de j ← 1 à 4 fazer
        se (T[i,j] > max) então
            max ← T[i,j]
        fim se
    fim de
fim de
```

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Para procurarmos a posição do menor elemento de uma tabela, agimos da mesma forma que ao procurar o menor elemento, ou seja:
  - Começamos por supor que a primeira posição (índice) é a posição do menor elemento da tabela;
  - Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o elemento da primeira posição com os restantes elementos da tabela;
  - Se um dos elementos for menor, então a posição deste, passa a ser a posição do novo menor.
  - Repetimos este processo até que já não haja novo menor.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Dada a seguinte tabela:**
  - T: tabela de inteiro[3, 4]
- **Procuremos a posição do mínimo:**

```
linha← 1
coluna ← 1
de i ← 1 à 3 fazer
    de j ← 1 à 4 fazer
        se (T[i, j] < T[linha, coluna]) então
            linha← i
            coluna ← j
        fim se
    fim de
fim de
```

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- De mesmo, para procurarmos a posição do maior elemento de uma tabela,
  - Começamos por supor que a primeira posição é a posição do maior elemento da tabela.
  - Em seguida percorremos a tabela usando uma instrução repetitiva, comparando o primeiro elemento com os restantes elementos da tabela.
  - Se um dos elementos for maior, então a posição deste passa a ser a posição do novo maior.
  - Repetimos este processo até que já não haja novo maior.

1. **Introdução**
2. **InSTRUçõEs  
executáveis  
básicas**
3. **Operadores**
4. **InSTRUçõEs  
condicionais**
5. **InSTRUçõEs  
repetitivas**
6. **Vectores**
7. **Algoritmos  
básicos para lidar  
com vectores**
8. **Matrizes**
9. **Algoritmos  
básicos para lidar  
com matrizes**
10. **Funções e  
procedimentos**

- **Dada a seguinte tabela:**
  - T: tabela de inteiro[3, 4]
- **Procuremos a posição do máximo:**

```
linha← 1
coluna ← 1
de i ← 1 à 3 fazer
    de j ← 1 à 4 fazer
        se (T[i,j] > T[linha, coluna]) então
            linha← i
            coluna ← j
        fim se
    fim de
fim de
```

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Ordenar uma tabela consiste em organizar os seus elementos em ordem crescente, descrente ou numa outra ordem específica.
- Existem vários tipos de algoritmo de ordenação e nós vamos aprender o mais simples dentre eles
  - Algoritmo de ordenação à bolha (Bubble sort algorithm).

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

O algoritmo de ordenamento à bolha tem o seguinte princípio:

- A ideia é percorrer o vector diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência. (Wikipedia)

Passo a passo:

- A gente percorre a tabela;
- Compara cada par de elementos vizinhos;
- Se o par não estiver ordenado, a gente intercambia eles de modo á ordená-los.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- Tomemos a seguinte tabela não ordenada como exemplo:

14	33	27
35	10	9

- Para ordená-la em ordem crescente usando o algoritmo de ordenação à bolha, começamos com os primeiros dois elementos, comparando-os para verificar qual deles é o maior.
- Neste caso, o valor 33 é maior que 14, portanto, ele já está classificado:

14	33	27
35	10	9

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Em seguida, comparamos 33 com 27:

14	33	27
35	10	9

- Verificamos que 33 é maior que 27 e esses dois valores devem ser intercambiados:

14	33	27
35	10	9

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

A nova tabela fica assim:

14	27	33
35	10	9

Em seguida, comparamos 33 e 35. Verificamos que estão em posições já classificadas:

14	27	33
35	10	9

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

Então nos movemos para os próximos dois valores, 35 e 10:

14	27	33
35	10	9

Verificamos que 35 é maior que 10. E que eles não estão ordenados:

14	27	33
35	10	9

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

A nova tabela fica assim:

14	27	33
10	35	9

Em seguida, comparamos 35 e 9:

14	27	33
10	35	9

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Verificamos que 35 é maior que 9. E que eles não estão ordenados:

14	27	33
10	35	9

Intercambiaremos esses valores: E verificamos que chegamos ao final da tabela. Depois de uma iteração, a tabela ficará assim:

14	27	33
10	9	35

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Depois da segunda iteração, a tabela ficará assim:

14	27	10
9	33	35

Observe que depois de cada iteração, pelo menos um elemento “sobe para cima”:

14	10	9
27	33	35

10	9	14
27	33	35

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

E quando não for mais necessário intercambiar elementos, o algoritmo supõe que a tabela está completamente ordenada:

9	10	14
27	33	35

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Dada a seguinte tabela:**
  - `T: tabela de inteiro[2,3] ← {{14, 33, 27}, {35, 10, 9}}`
- **Para classificá-la em ordem crescente procedemos da seguinte forma:**

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

```
M ← 3 // Número de linhas
N ← 4 // Número de colunas
de i ← 1 à M-1 fazer
    de j ← 1 à N-1 fazer
        de k ← i+1 à M fazer
            de l ← j+1 à N fazer
                se (T[i,j] > T[k,l]) então
                    temp ← T[i,j]
                    T[i,j] ← T[k,l]
                    T[k,l] ← temp
                fim se
            fim de
        fim de
    fim de
fim de
```



# 10. Funções e Procedimentos

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

- Voltemos à nossa analogia com a receita de culinária.
  - Recorde-se que comparamos o algoritmo à uma receita de culinária onde de antemão estão definidos os ingredientes e os passos necessários para realizar um prato.
  - Imaginemos um restaurante cujo menu tem os seguintes pratos com componentes repetidos:
    - Arroz com feijão,
    - Massa com feijão,
    - Arroz com peixe,
    - Massa com peixe.

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- Se o chefe cozinheiro do restaurante for preguiçoso, como todos os programadores, em vez de fazer a receita de todos estes pratos repetindo os passos necessários para realizar cada componente, receita após receita...
- Fazia receitas separadas dos pratos singulares (arroz, massa, feijão e peixe) e os referenciava nas receitas dos pratos compostos.

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Exemplo:

- **Receitas dos componentes:**

- Receita Arroz;
- Receita Feijão;
- Receita Peixe;
- Receita Massa;

Receita  
Arroz

Receita  
Feijão

Receita  
Peixe

ReceitaM  
assa

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

- **Receitas principais**
  - **Receita Arroz com Feijão**
    - Ver Receita Arroz
    - Ver Receita Feijão
  - **Receita Massa com Feijão**
    - Ver Receita Massa
    - Ver Receita Feijão

Receita Arroz com Feijão

[Ver receita Arroz](#)

[Ver receita Feijão](#)

Receita Massa com Peixe

[Ver receita Massa](#)

[Ver receita Peixe](#)

- 1. **Introdução**
- 2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
- 3. **Operadores**
- 4. **InSTRUÇÕES CONDICIONAIS**
- 5. **InSTRUÇÕES REPETITIVAS**
- 6. **Vectores**
- 7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
- 8. **Matrizes**
- 9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
- 10. **Funções e procedimentos**

- Em programação, o acto de criar pequenas receitas dos componentes que se repetem chama-se criar funções ou procedimentos;
- E o acto de referenciar as pequenas receitas na recita principal designa-se chamar ou executar a função ou o procedimento.

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

- **Definição:** Funções e Procedimentos são subprogramas escritos fora do programa principal mas que se executam dentro deste.
- Diferentemente de um procedimento, uma função sempre retorna um valor.

- 1. **Introdução**
- 2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
- 3. **Operadores**
- 4. **InSTRUÇÕES CONDICIONAIS**
- 5. **InSTRUÇÕES REPETITIVAS**
- 6. **Vectores**
- 7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
- 8. **Matrizes**
- 9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
- 10. **Funções e procedimentos**

- Ao criarmos uma função podemos deixar em aberto certas variáveis para serem fornecidas no seu chamamento.
- Voltemos à nossa receita de culinária, suponhamos que está a fazer uma função para cozinhar feijão. Poderá deixar em aberto a quantidade e o tipo de feijão para ser especificado no acto do chamamento da função.

1. **Introdução**
2. **InSTRUÇÕES EXECUTÁVEIS BÁSICAS**
3. **Operadores**
4. **InSTRUÇÕES CONDICIONAIS**
5. **InSTRUÇÕES REPETITIVAS**
6. **Vectores**
7. **Algoritmos BÁSICOS PARA LIDAR COM VECTORES**
8. **Matrizes**
9. **Algoritmos BÁSICOS PARA LIDAR COM MATRIZES**
10. **Funções e procedimentos**

- **Essas variáveis deixadas em aberto para serem especificadas no chamamento da função ou do procedimento chamam-se parâmetros ou argumentos.**

1. **Introdução**
2. **InSTRUçõEs  
executáveis  
básicas**
3. **Operadores**
4. **InSTRUçõEs  
condicionais**
5. **InSTRUçõEs  
repetitivas**
6. **Vectores**
7. **Algoritmos  
básicos para lidar  
com vectores**
8. **Matrizes**
9. **Algoritmos  
básicos para lidar  
com matrizes**
10. **Funções e  
procedimentos**

## Sintaxe (Função)

Função <Nome da função> (<lista de parâmetros>) retorna <Tipo de variável>

Variáveis:

<Declaração das variáveis da função>

    Início

<Bloco de instruções executáveis>

    Retorna Expressão

    Fim

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- A <lista de parâmetros> tem a seguinte forma:

```
var1: Tipo, var2: Tipo, ... , varN: Tipo
```

## Exemplo:

Função Multiplicacao(A: Inteiro, B: Inteiro) retorna Inteiro

Variáveis:

```
produto: Inteiro
```

Início

```
produto ← A × B
```

```
retorna produto
```

Fim

- 1. **Introdução**
- 2. **InSTRUções executáveis básicas**
- 3. **Operadores**
- 4. **InSTRUções condicionais**
- 5. **InSTRUções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

## Sintaxe (Procedimento)

Função <nome do procedimento> (<lista de parâmetros>) retorna Vazio

Variáveis:

<Declaração das variáveis do procedimento>

Início

<Bloco de instruções>

Fim

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Exemplo:

Função Adicao(A: Inteiro, B: Inteiro) retorna Vazio

Variáveis: soma

Início

    soma ← A + B

    escrecer("A soma é ", soma)

Fim

## Obs:

- Um procedimento é uma função que não retorna nenhum valor;

- 1. **Introdução**
- 2. **Instruções executáveis básicas**
- 3. **Operadores**
- 4. **Instruções condicionais**
- 5. **Instruções repetitivas**
- 6. **Vectores**
- 7. **Algoritmos básicos para lidar com vectores**
- 8. **Matrizes**
- 9. **Algoritmos básicos para lidar com matrizes**
- 10. **Funções e procedimentos**

Embora declaradas fora do programa principal - geralmente antes do bloco de instruções do programa principal - as funções e procedimentos só são executados quando são chamados no bloco de instruções executáveis do programa principal.

1. Introdução
2. Instruções executáveis básicas
3. Operadores
4. Instruções condicionais
5. Instruções repetitivas
6. Vectores
7. Algoritmos básicos para lidar com vectores
8. Matrizes
9. Algoritmos básicos para lidar com matrizes
10. Funções e procedimentos

- **Chamar uma função:** Consiste em referenciar ou invocar uma função previamente definida.

## Sintaxe:

- <Nome da função> (<lista de valores para os parâmetros>)
- <Nome do procedimento> (<lista de valores para os parâmetros>)

1. **Introdução**
2. **InSTRUções executáveis básicas**
3. **Operadores**
4. **InSTRUções condicionais**
5. **InSTRUções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

**Obs:**

- **Ao chamar uma função ou procedimento a lista de valores para os parâmetros deve ter a seguinte forma:**
  - <Nome da função ou procedimento> (val1, val2, ..., valN)

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

## Exemplo:

```
Algoritmo Soma_de_dois_numeros
```

### Variáveis:

```
    X, Y, Z: Inteiro
```

```
    // Declarando a função
```

```
    Função Soma(A: Inteiro, B: Inteiro) retorna
    Inteiro
```

### Variáveis:

```
    resultado: Inteiro
```

```
    A, B: Inteiro
```

```
    Início
```

```
        resultado ← A + B
```

```
        retorna resultado
```

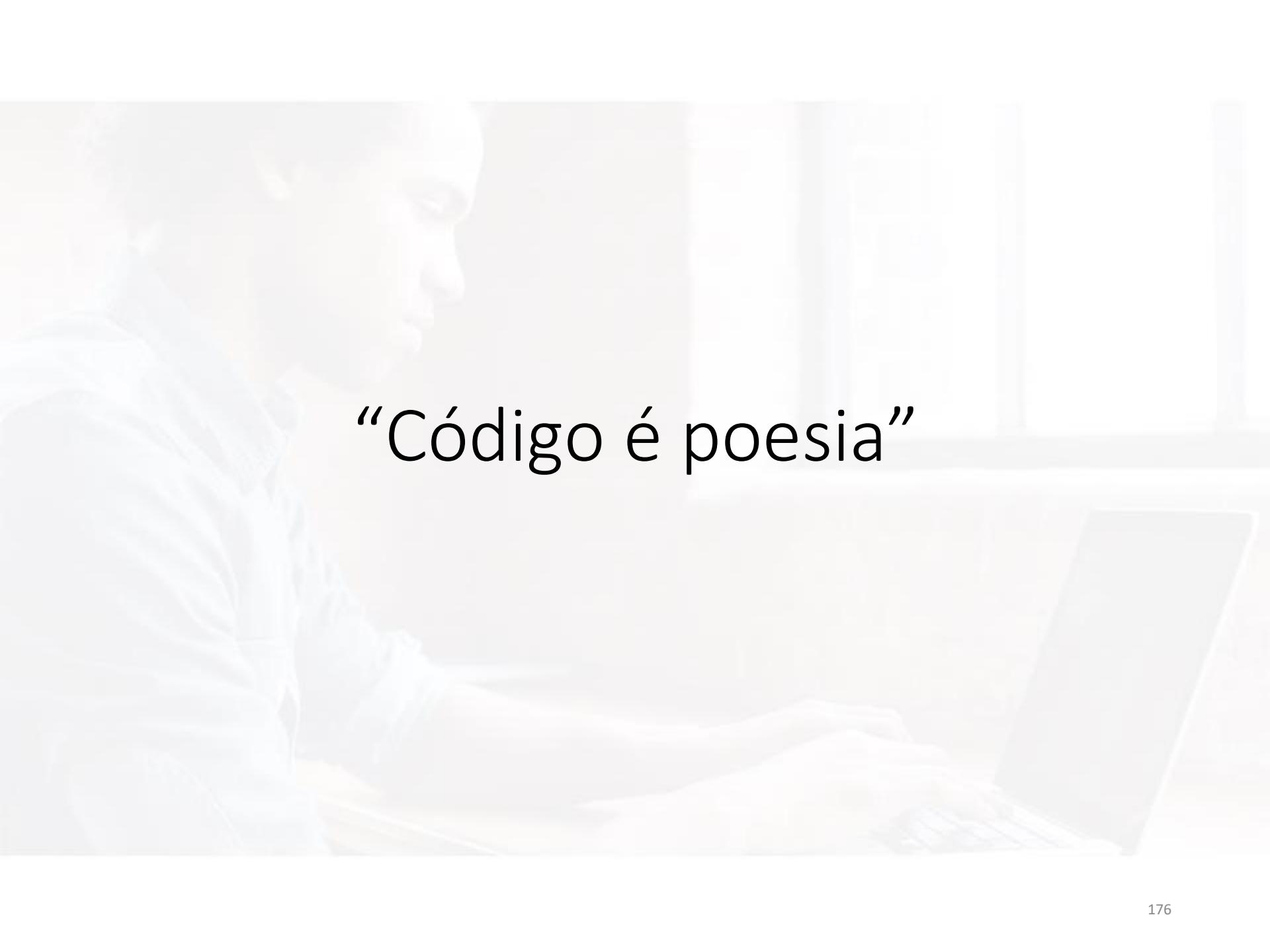
```
    Fim
```

1. **Introdução**
2. **Instruções executáveis básicas**
3. **Operadores**
4. **Instruções condicionais**
5. **Instruções repetitivas**
6. **Vectores**
7. **Algoritmos básicos para lidar com vectores**
8. **Matrizes**
9. **Algoritmos básicos para lidar com matrizes**
10. **Funções e procedimentos**

Início

```
    Escrever("Bem-vindo a aplicação soma");  
    Escrever("Digite o primeiro número");  
    Ler(X);  
    Escrever("Digite o segundo número");  
    Ler(Y);  
    Z ← Soma(X, Y); // Chamar a função Soma  
    Escrever("A soma é ", Z);
```

Fim



“Código é poesia”