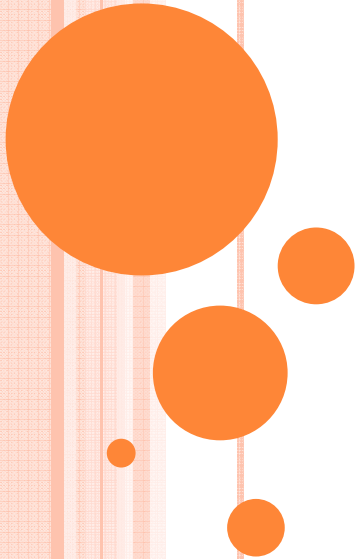


ESTRUTURAS DE REPETIÇÃO E VARIÁVEIS INDEXADAS.

Prof. Me. Hélio Esperidião

Definição

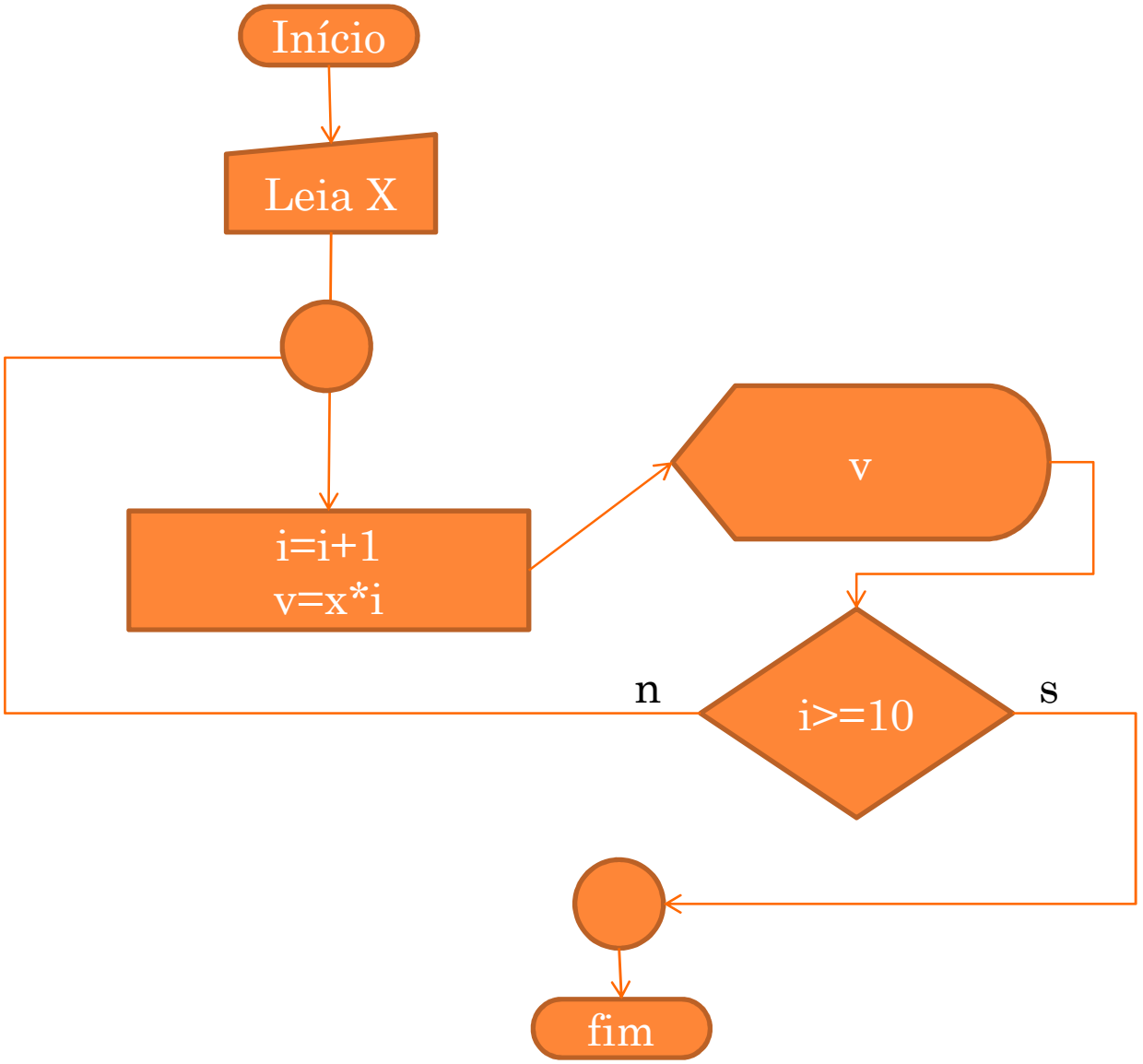
- As estruturas de repetição – também conhecidas como estruturas de looping – permitem que você executa uma linha ou bloco de código repetidamente até que uma condição seja verdadeira.

- As estruturas de repetições são muito importantes para programas, pois muitas vezes o mesmo procedimento deve que ser executado mais de uma vez.

Definição

- Basicamente existem três tipos de estrutura de repetição: para(FOR), enquanto(WHILE) e faça enquanto (DO WHILE).

Fluxograma - Repetição



Estrutura de repetição – Para (FOR)

- O FOR como qualquer iteração precisa de uma variável para controlar a quantidade de loops (voltas,vezes).

- Em for, essa variável deverá ser inicializada, indicada seu critério de execução, e forma de incremento ou decremento.

Estrutura de repetição – Para (FOR)

- For precisa de três condições.
- Condição de inicio.
- Condição de término.
- Condição de incremento.

PORTUGUÊS ESTRUTURADO

```
algoritmo "semnome"
var
i: inteiro
x: inteiro
n:inteiro
inicio

escreva("Digite um número ")
leia(x)

para i de 1 ate 10 passo 1 faca
      n<-x*i
      escreval(n)
fimpara


// Seção de Comandos
fimalgoritmo
```

i começa com 1

i vai crescendo de 1 em 1

A variável **i** vai até 10

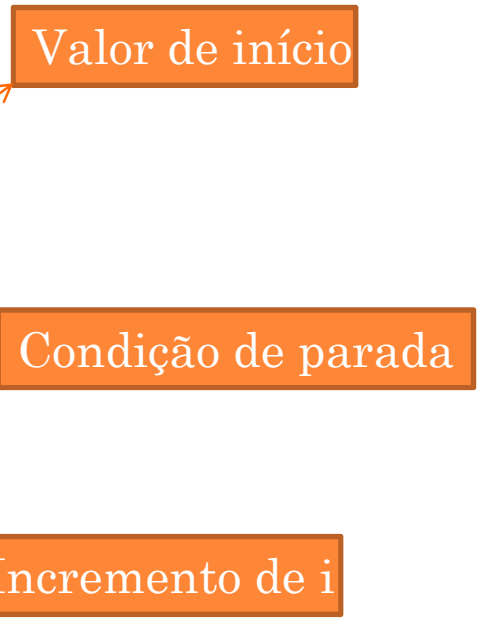
Repete até que **i** seja igual a 10

Estrutura de repetição – enquanto(while)

- Enquanto precisa de três condições.
- Condição de inicio.
- Condição de término.
- Condição de incremento.
- Caso alguma destas condições seja esquecida o programa pode ficar em *loop* infinito

Estrutura de repetição – enquanto(while)

```
algoritmo "semnome"
var
 i: inteiro
 x: inteiro
 v: inteiro
inicio

leia(x)
i<-1
enquanto i<=10 faca
    v<-i*x
    escreval(v)
    i<-i+1
fimenquanto

fimalgoritmo
```



Loop Infinito

```
algoritmo "semnome"
var
  i: inteiro
  x: inteiro
  v: inteiro
inicio

leia(x)
i<-1
enquanto i<=10 faca
     v<-i*x
     escreval(v)
fimenquanto

fimalgoritmo
```

Encontre o problema!

Loop infinito

Estrutura de repetição – Repita(do while)

- Repita precisa de três condições.
- Condição de inicio.
- Condição de término.
- Condição de incremento.
- Caso alguma destas condições seja esquecida o programa pode ficar em *loop* infinito
- O repita é sempre executado pelo menos uma vez

PORTUGUÊS ESTRUTURADO

```
algoritmo "semnome"
var
 i: inteiro
 r: inteiro
 n: inteiro
inicio

leia(n)

i<-1

repita
    r<-i*n
    i<-i+1
    escreval(r)
ate(i>10)

fimalgoritmo
```

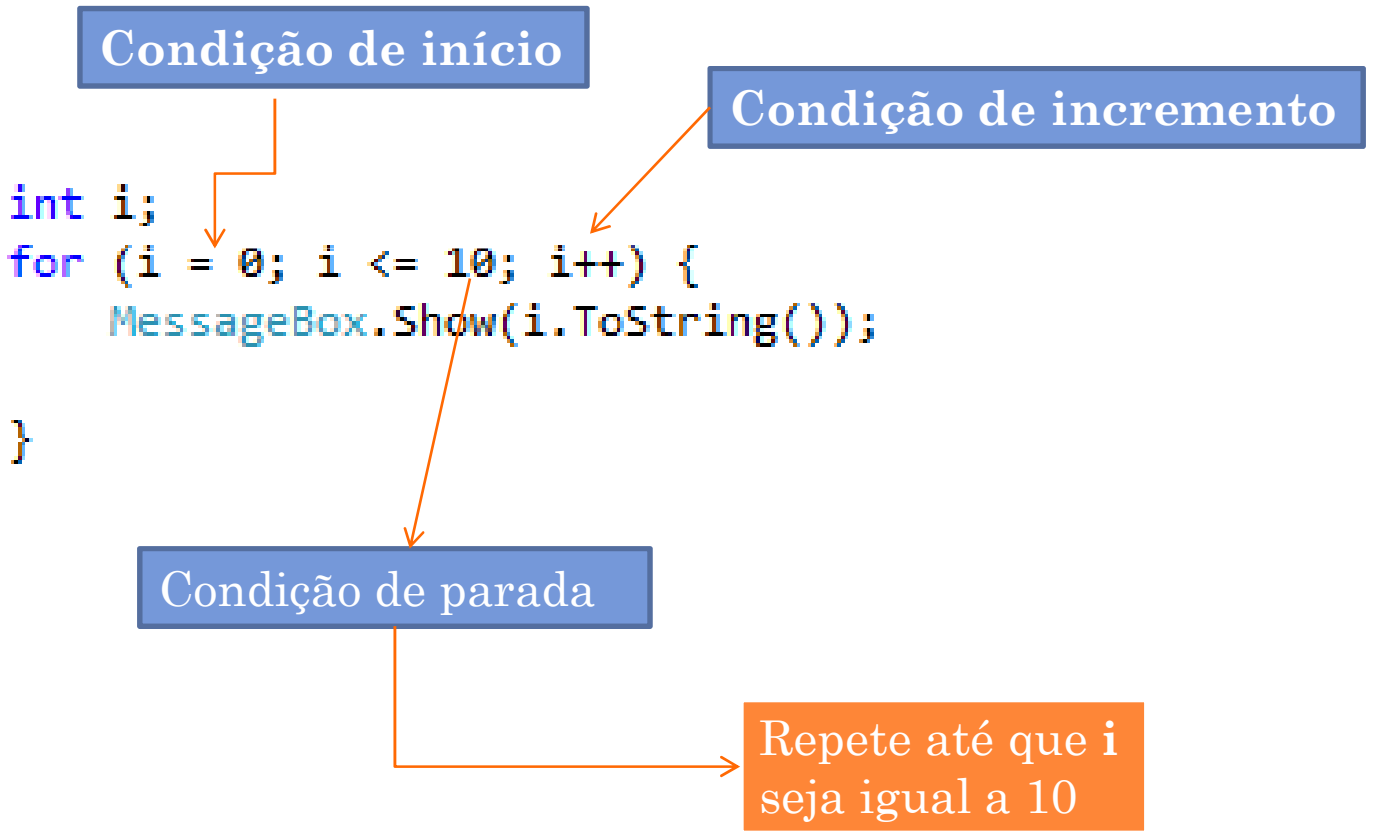
Valor de início

Incremento de i

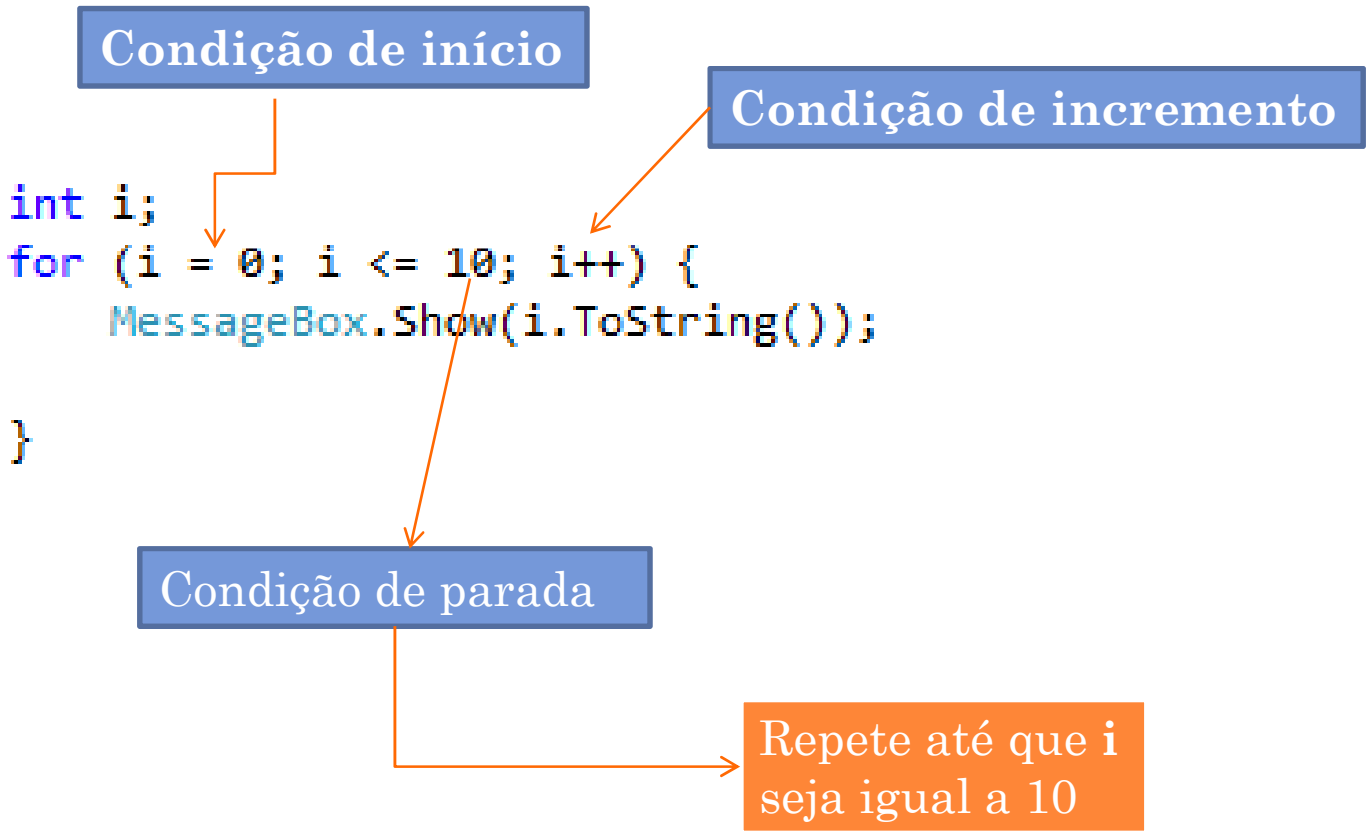
Condição de parada

O repita testa no final. Obrigatoriamente o código Será executado pelo menos uma vez

C# - FOR


Condição de início
Condição de incremento
int i;
for (i = 0; i <= 10; i++) {
MessageBox.Show(i.ToString());
}
Condição de parada
Repete até que i seja igual a 10

C# - FOR


Condição de início
Condição de incremento
int i;
for (i = 0; i <= 10; i++) {
MessageBox.Show(i.ToString());
}
Condição de parada
Repete até que i seja igual a 10

C# - WHILE

Repete até que **i** seja igual a 10

Condição de início

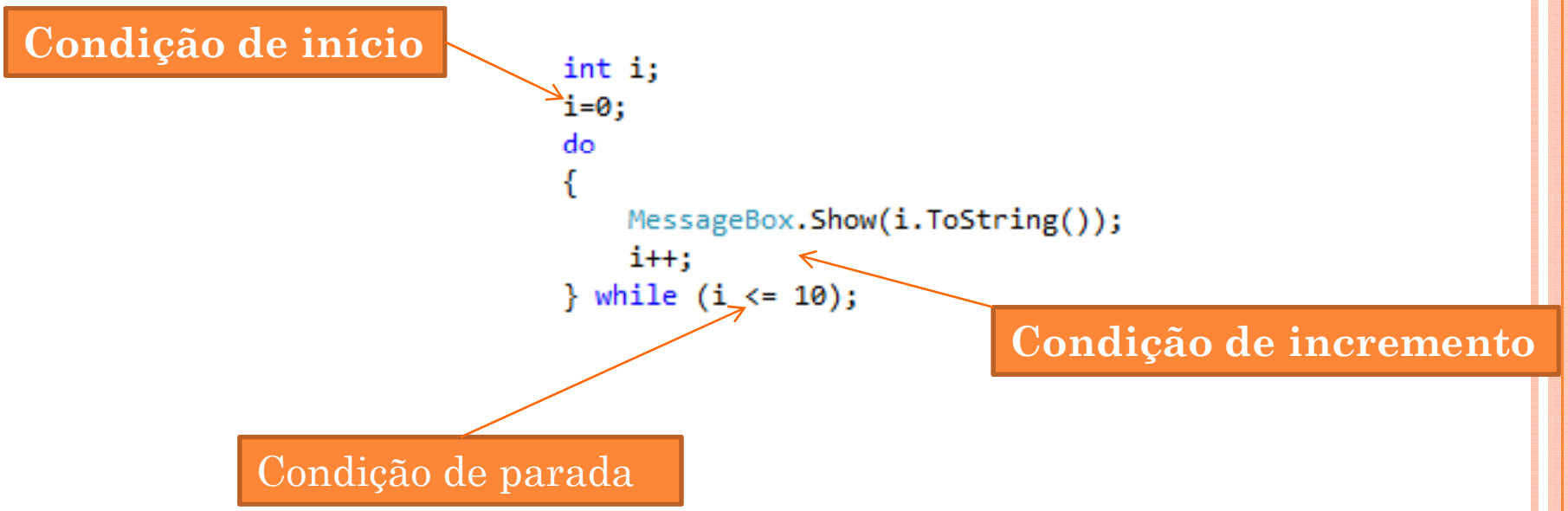
Condição de parada

```
int i;
i=0;
while(i<=10){
    MessageBox.Show(i.ToString());
    i++;
}
```

Condição de incremento

C# - DO...WHILE

- Do...while executa pelo menos uma vez, pois o teste é feito no final.



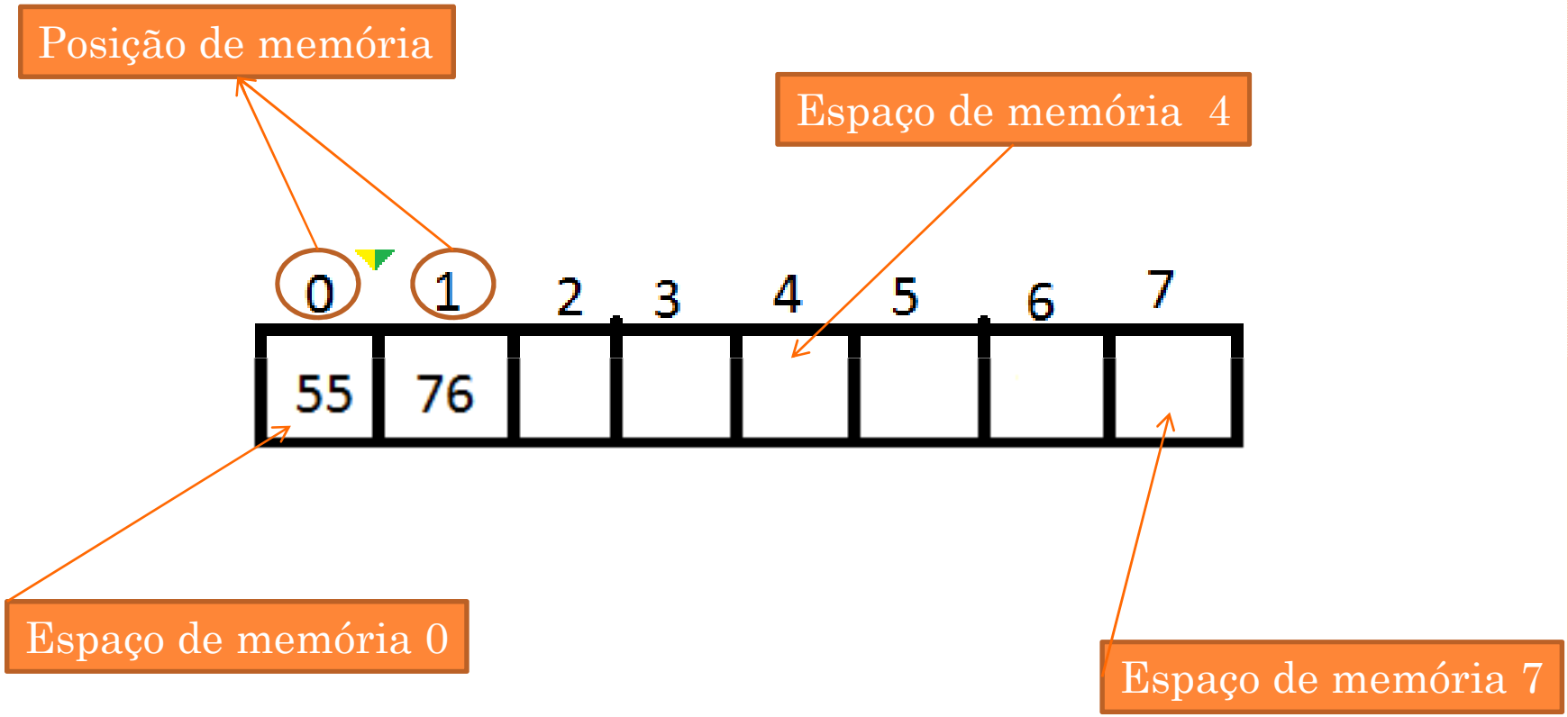
VETORES

- Vetores são **Estruturas de Dados** capazes de armazenar uma quantidade ***pré-definida*** de valores ***do mesmo tipo.***

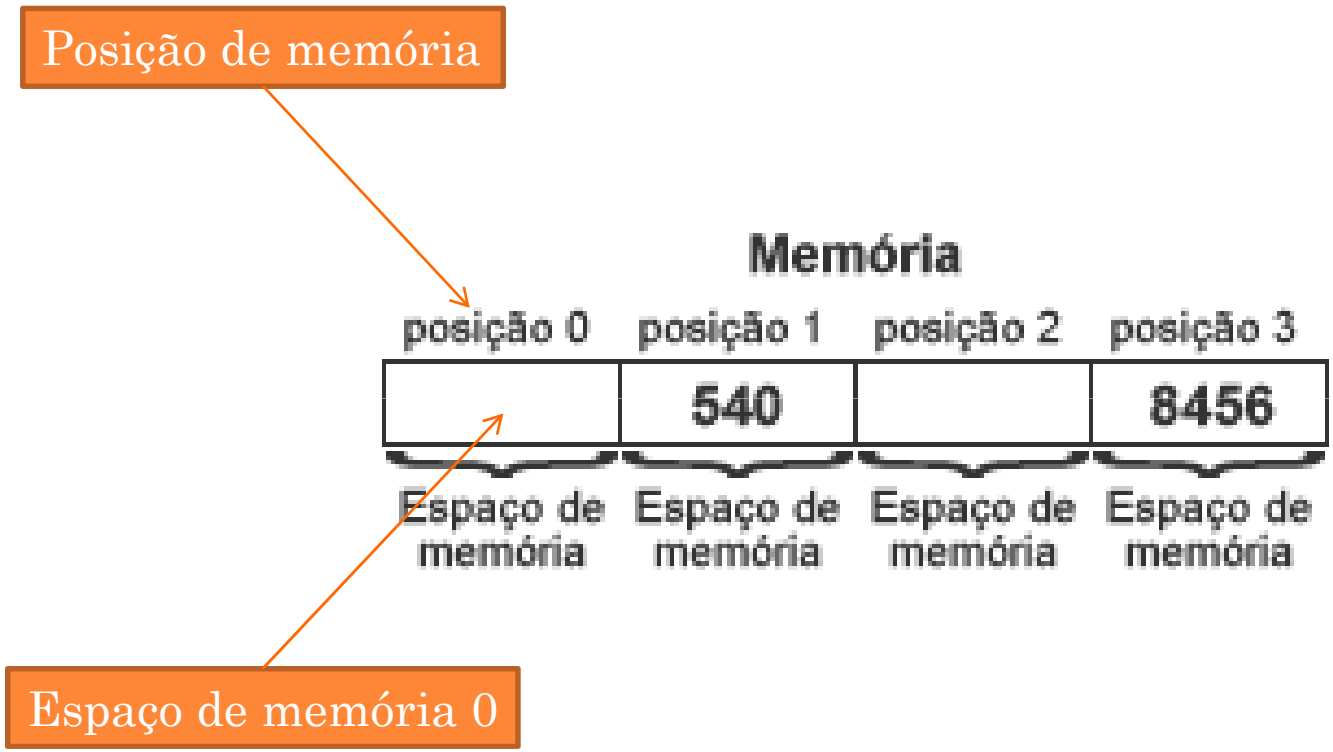
- Permitem que sejam armazenados diversos valores em uma mesma variável.

- Vetores são unidimensionais.

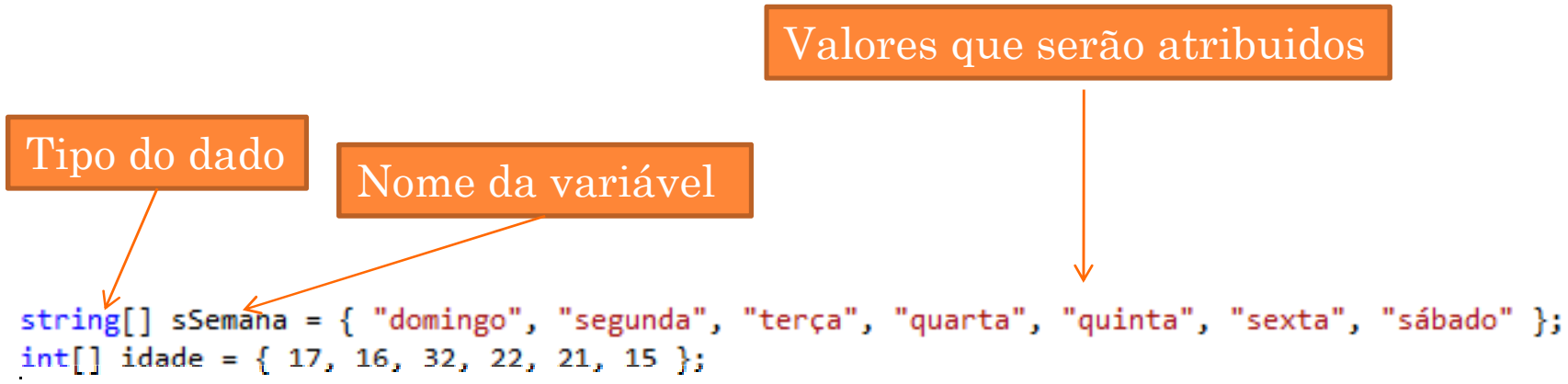
VETOR


Posição de memória
Espaço de memória 4
0
1
2
3
4
5
6
7
55
76
Espaço de memória 0
Espaço de memória 7

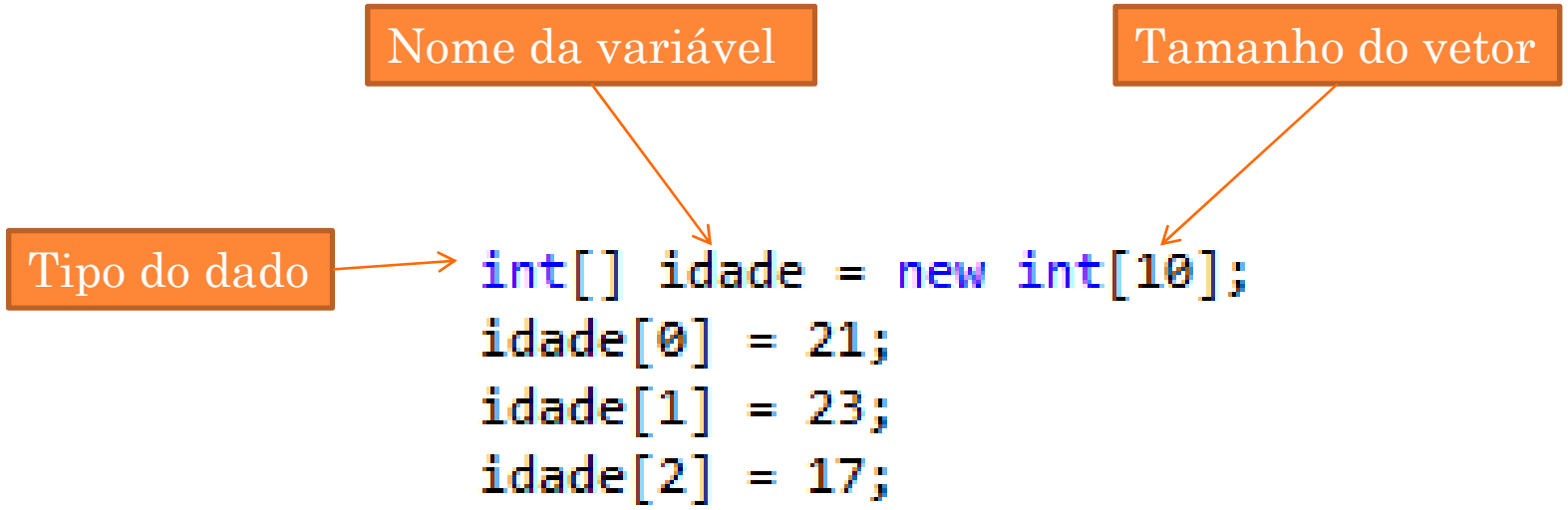
VETORES


Posição de memória
Memória
posição 0
posição 1
posição 2
posição 3
540
8456
Espaço de memória
Espaço de memória
Espaço de memória
Espaço de memória
Espaço de memória 0

Declarando vetores



Declarando vetores

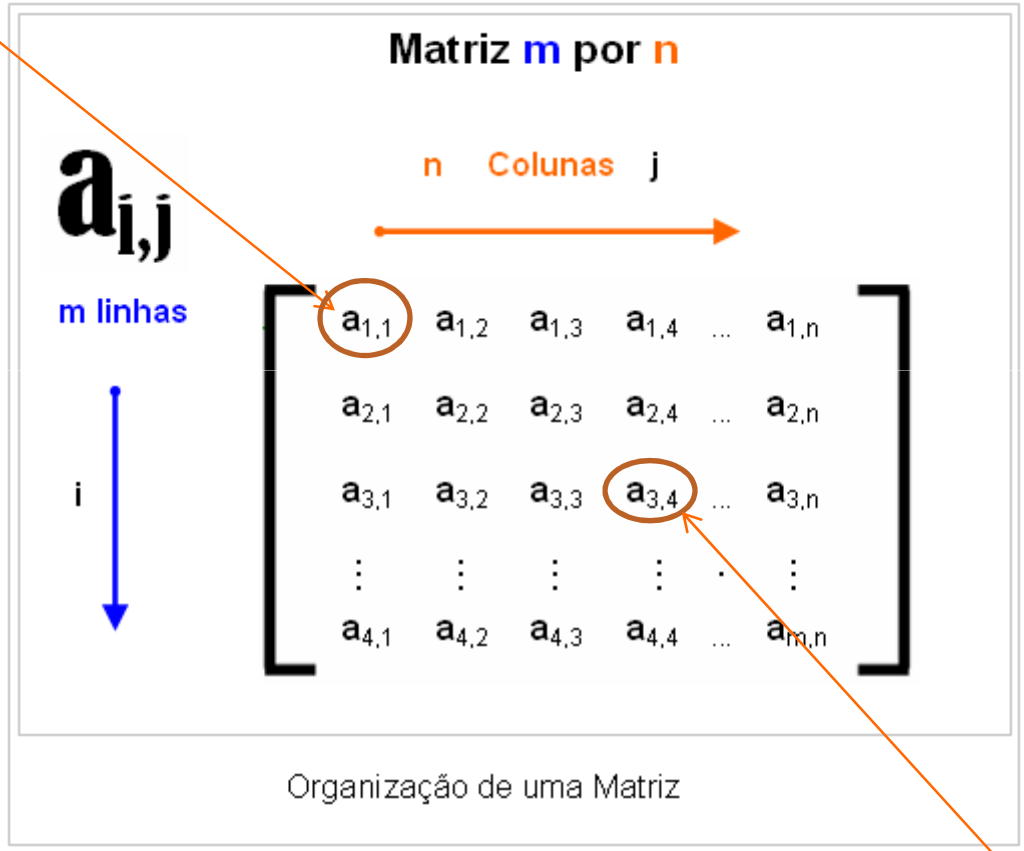


Matrizes

- São **Estruturas de Dados** capazes de armazenar uma quantidade ***pré-definida*** de valores ***do mesmo tipo.***
- Permitem que sejam armazenados diversos valores em uma mesma variável.
- Vetores são unidimensionais.

MATRIZ

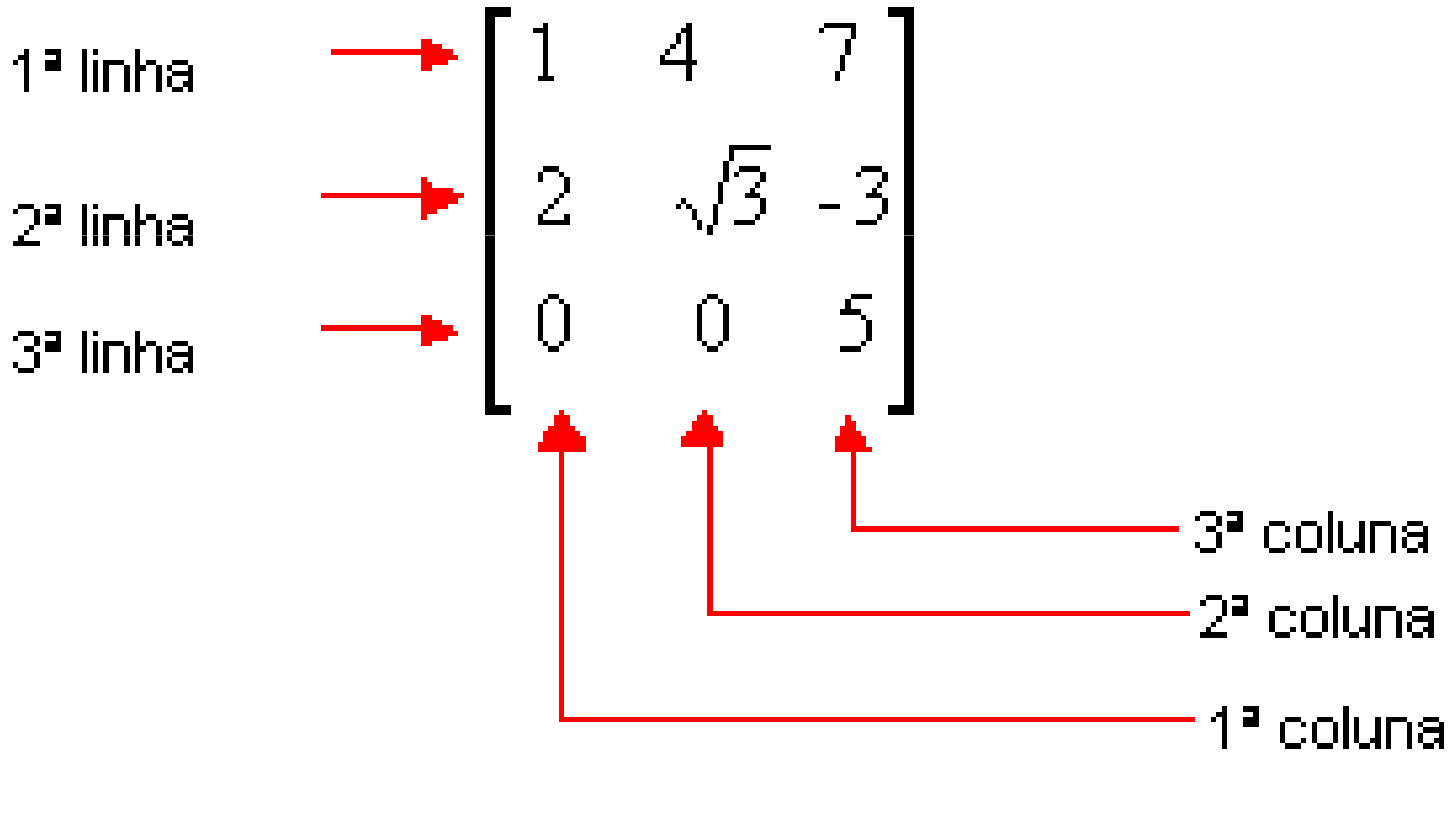
Posição de memória (1,1) – Linha 1 e Coluna 1



Organização de uma Matriz

Posição de memória (3,4) – Linha 3 e Coluna 4

VETORES

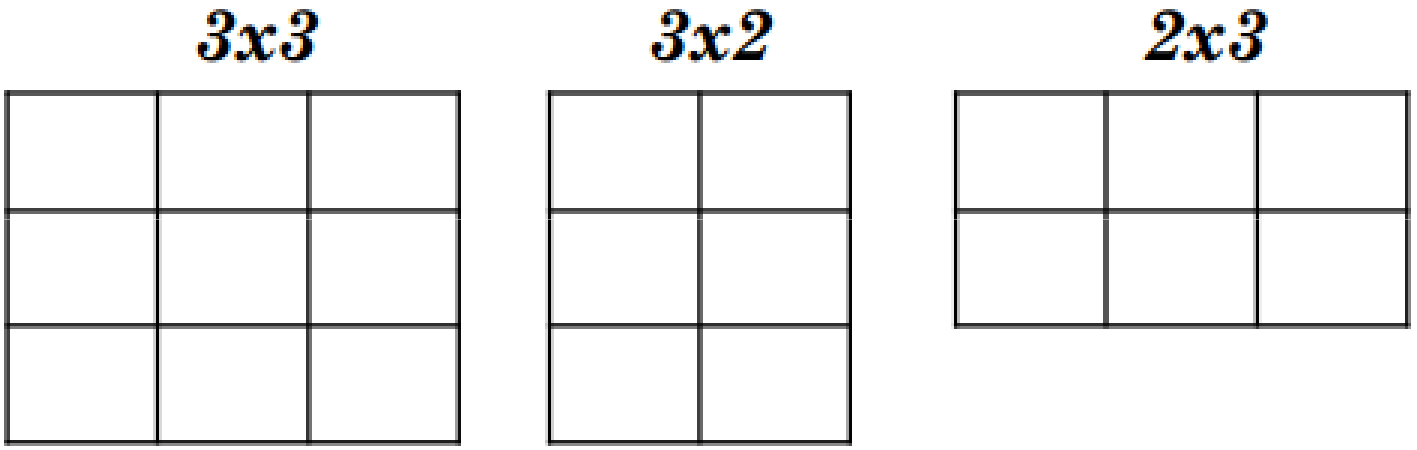

1ª linha
2ª linha
3ª linha
1 4 7
2 √3 -3
0 0 5
3ª coluna
2ª coluna
1ª coluna

Matrizes

- As matrizes são, comumente referenciadas através de suas dimensões (quantidade de linhas e colunas).

- A notação comum é: MxN, onde:
  - M é a dimensão vertical (quantidade de linhas)
  - N é dimensão horizontal (quantidade de colunas)

VETORES



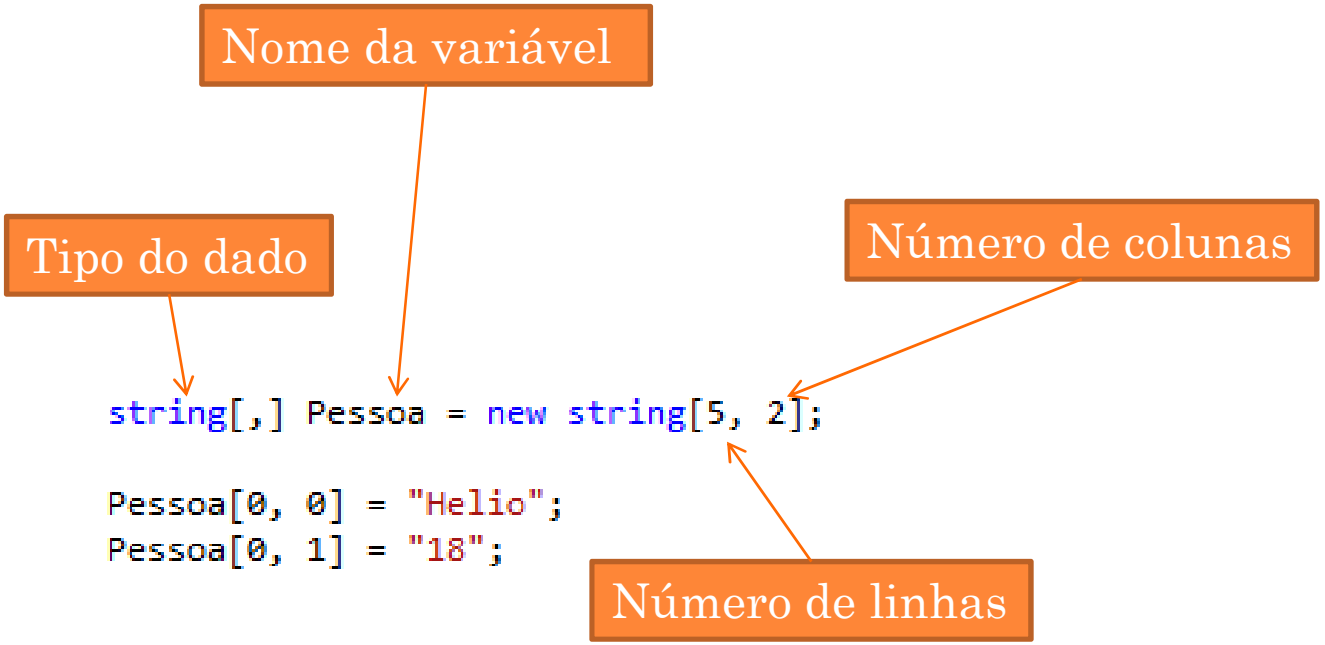
Linhas x Colunas

Matriz

```
algoritmo "semnome"
var
produtos: vetor [1..3, 1..2] de caractere

inicio
produtos[1,1]<- "Refrigerante"
produtos[1,2]<- "1.8"

produtos[2,1]<- "Leite"
produtos[2,2]<- "2.8"

produtos[3,1]<- "Laranja"
produtos[3,2]<- "0.8"

escreval(produtos[1,1]," - ",produtos[1,2])

escreval(produtos[3,1]," - ",produtos[3,2])

fimalgoritmo
```

	1	2
1	Refrigerante	1.8
2	Leite	2.8
3	Laranja	0.8

Declarando matrizes




```
string[,] Pessoa = new string[5, 2];

Pessoa[0, 0] = "Helio";
Pessoa[0, 1] = "18";


Pessoa[1, 0] = "Andreia";
Pessoa[1, 1] = "32";
```