

CONSTRUÇÃO DE ALGORITMOS

Bacharelado em Sistemas da Informação

Prof. Marco André Abud Kappel

Aula 8 – Vetores

Vetores

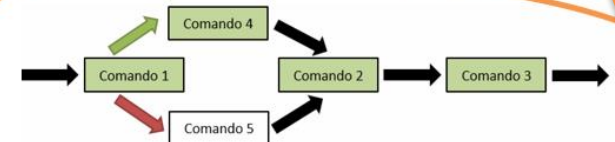
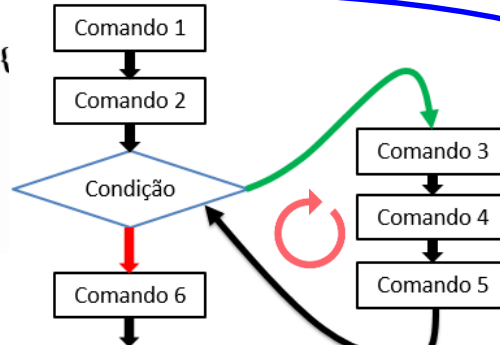
• Anteriormente:

Hoje:
Vetores

```
enquanto(condição){
  comando1
  comando2
  comando3
  ...
}
```

```
para(variávelDeControle = valorInicial; condição; variávelDeControle = novoValor){
  comando1
  comando2
  comando3
  ...
}
```

```
faça {
  comando1
  comando2
  comando3
  ...
} enquanto(condição)
```



```
se (condição){
  comando1
  comando2
  comando3
  ...
} senao {
  comando4
  comando5
  comando6
  ...
}
```



```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){
  bloco de comandos
  retorne algum_resultado
}
```

Parâmetros por valor

Parâmetros por referência



falso verdadeiro

nao e ou

Operador	Operação
==	igual
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

	x	y	x & y
falso	falso	falso	falso
falso	falso	verdadeiro	falso
falso	verdadeiro	falso	falso
falso	verdadeiro	verdadeiro	falso
verdadeiro	falso	falso	falso
verdadeiro	falso	verdadeiro	falso
verdadeiro	verdadeiro	falso	falso
verdadeiro	verdadeiro	verdadeiro	verdadeiro

Vetores

- **Introdução**

- Toda linguagem de programação procura permitir que se trabalhe com **dados** e, também, que se possa armazená-los em um local de memória associado diretamente a um **identificador**.
- Dentre as possibilidades de estruturas que possam armazenar dados, já conhecemos uma:

Vetores

- **Introdução**

- Toda linguagem de programação procura permitir que se trabalhe com **dados** e, também, que se possa armazená-los em um local de memória associado diretamente a um **identificador**.
- Dentre as possibilidades de estruturas que possam armazenar dados, já conhecemos uma:
- A variável!

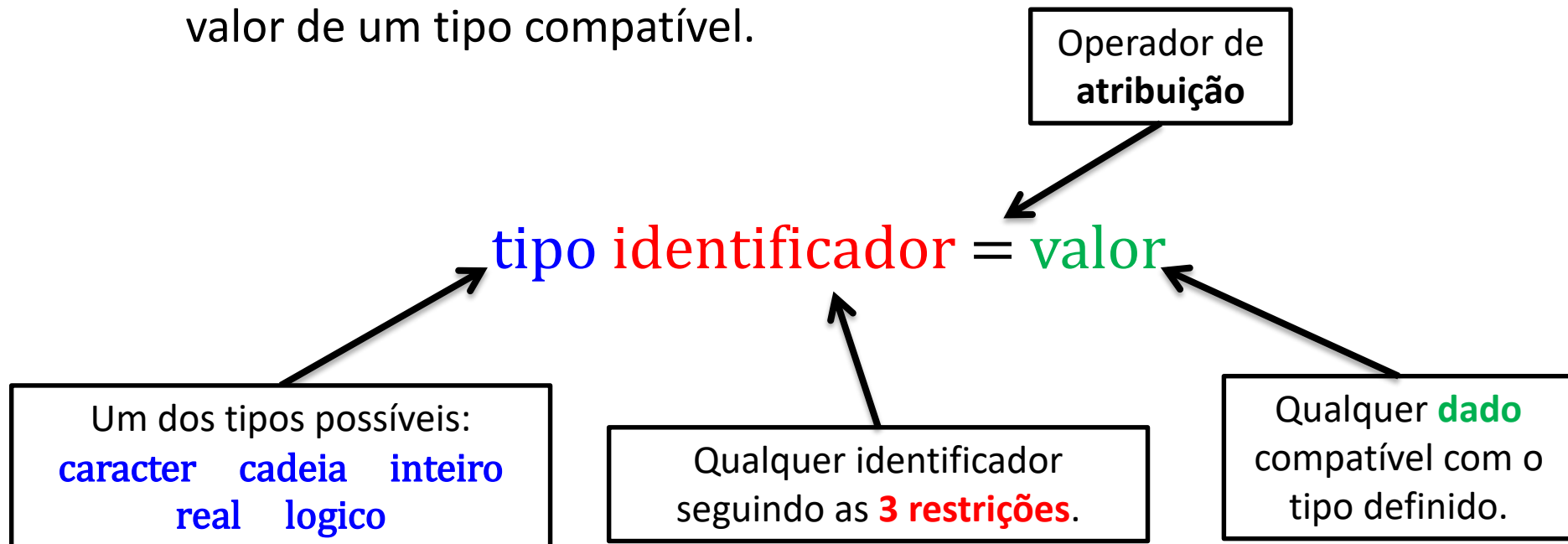
“algum
texto”



Vetores

- **Introdução**

- Vimos que, para criar uma variável, é necessário fazer uma **declaração**.
- Depois, podemos atribuir um **valor** à variável, desde que seja um valor de um tipo compatível.



Vetores

- Arranjos

- Veremos, agora, estruturas capazes de armazenar **vários valores**: os **arranjos**.
- Desta forma, podemos usar uma **única variável** para armazenar vários valores ao mesmo tempo.



Atenção! Todos os valores precisam ser do **mesmo tipo**!



Vetores

- Arranjos

- Estudaremos **dois tipos** de arranjos: o arranjo **unidimensional** e o arranjo **bidimensional**.

V =

99	17	16	13	14	20	19
----	----	----	----	----	----	----

Vetor

M =

41	31	21	51	61	21	31	41
11	11	11	11	11	11	11	11
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10
40	30	20	50	60	20	30	40

Matriz

Vetores

- Arranjos

- Estudaremos **dois tipos** de arranjos: o arranjo **unidimensional** e o arranjo **bidimensional**.

V =

99	17	16	13	14	20	19
----	----	----	----	----	----	----

Vetor

M =

41	31	21	51	61	21	31	41
11	11	11	11	11	11	11	11
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10
40	30	20	50	60	20	30	40

Matriz

Veremos agora

Vetores

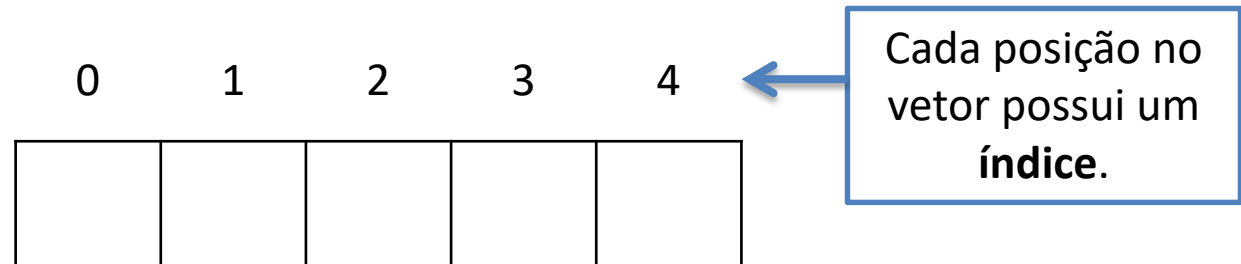
- **Estrutura**
 - Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

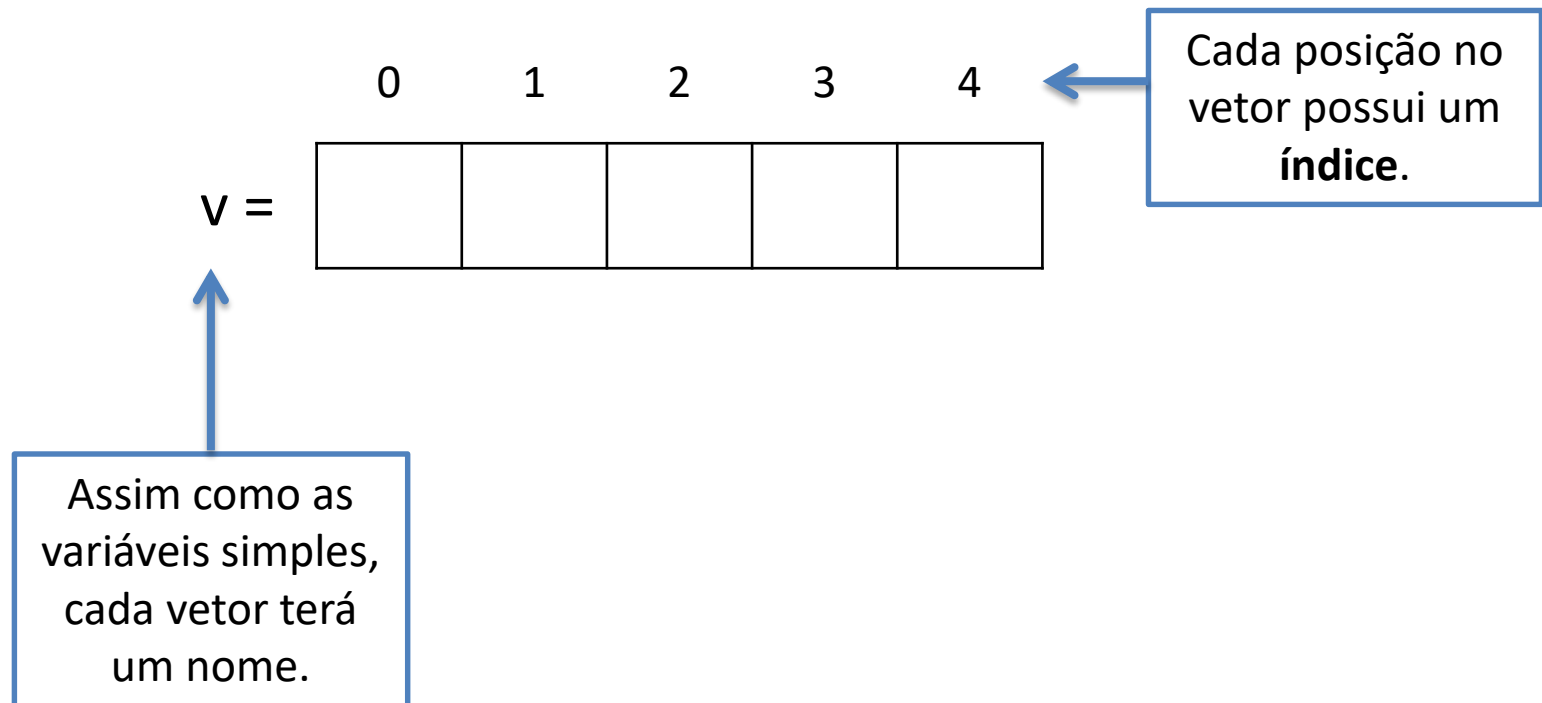
- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

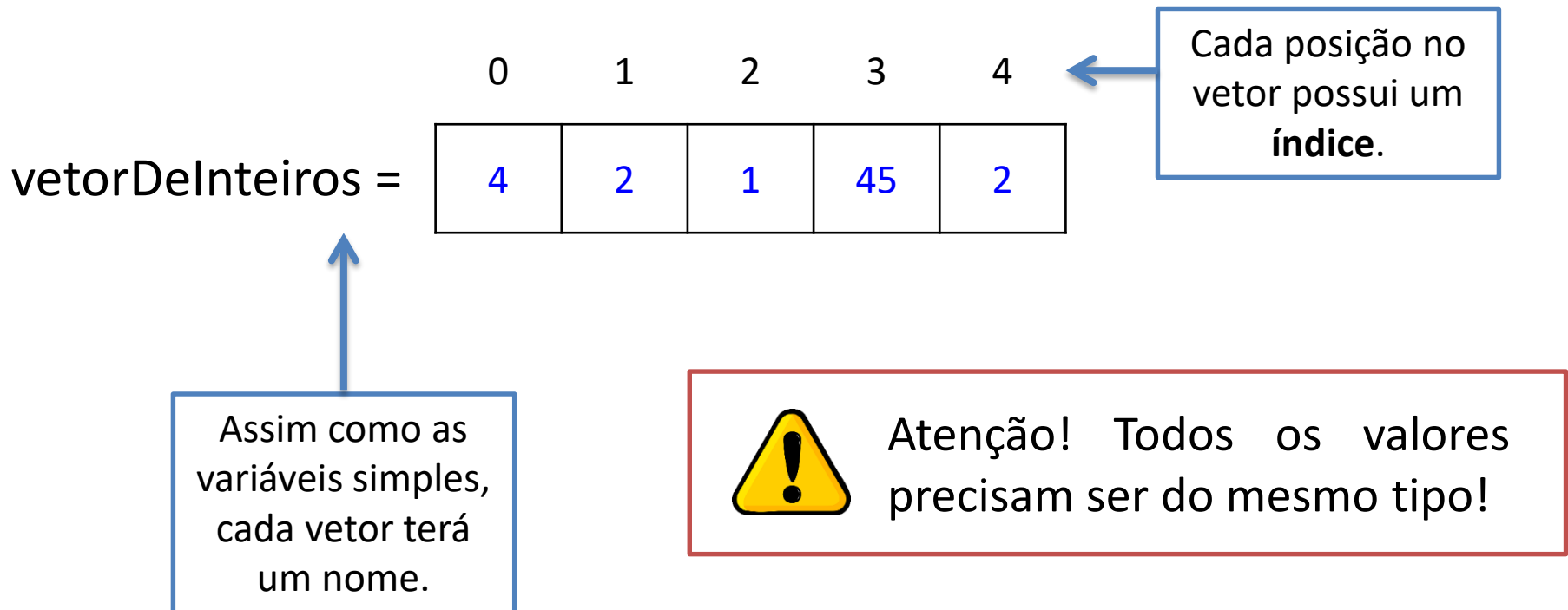
- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

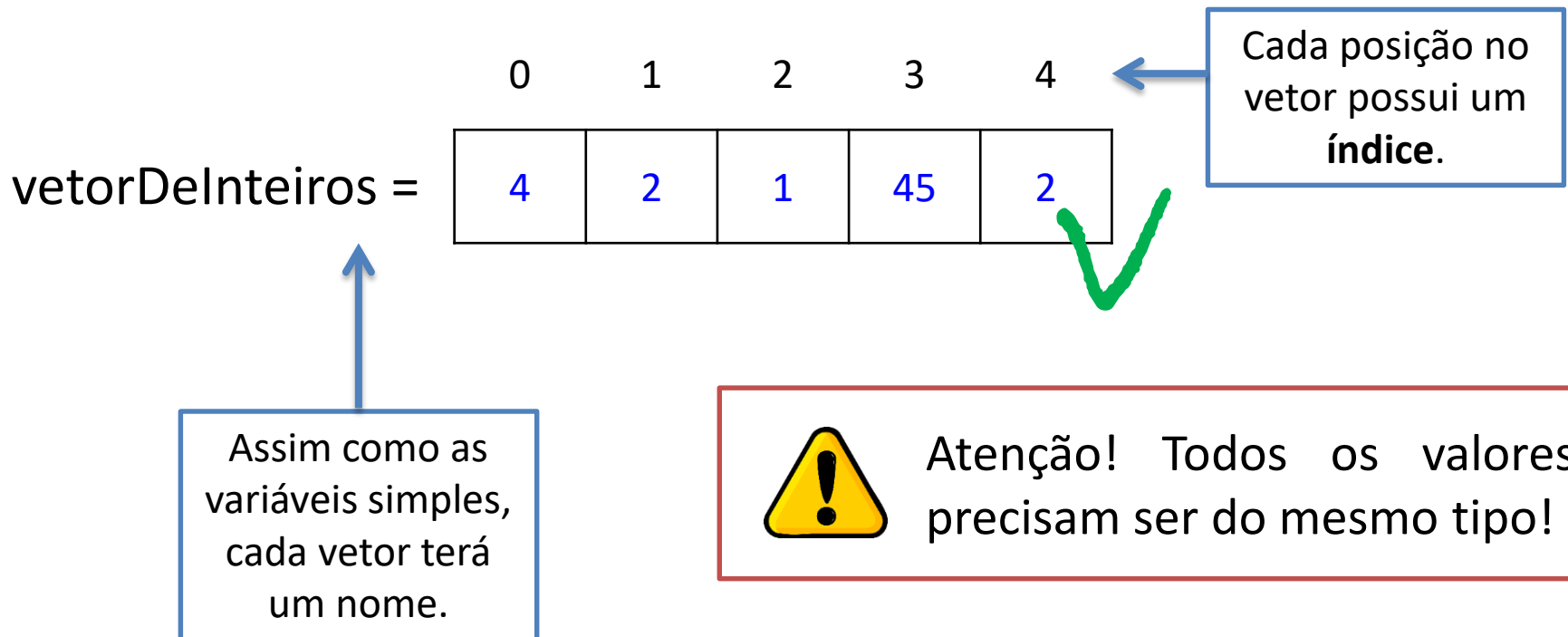
- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

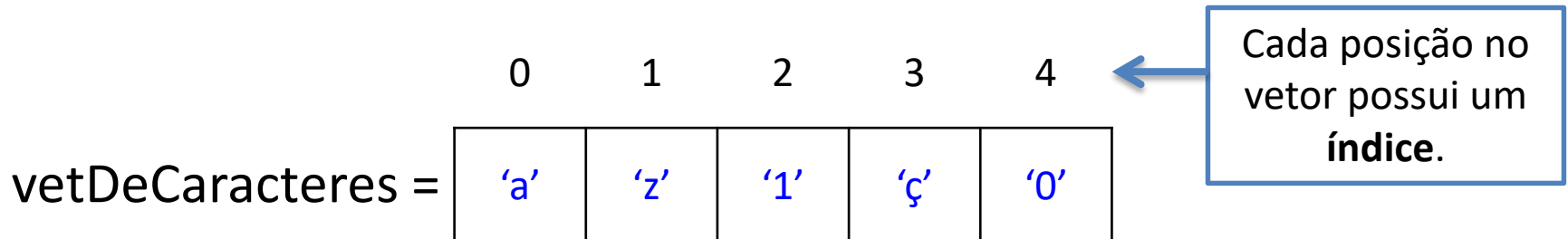
- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

- Um vetor é constituído por uma série de elementos em sequência.



Assim como as variáveis simples, cada vetor terá um nome.

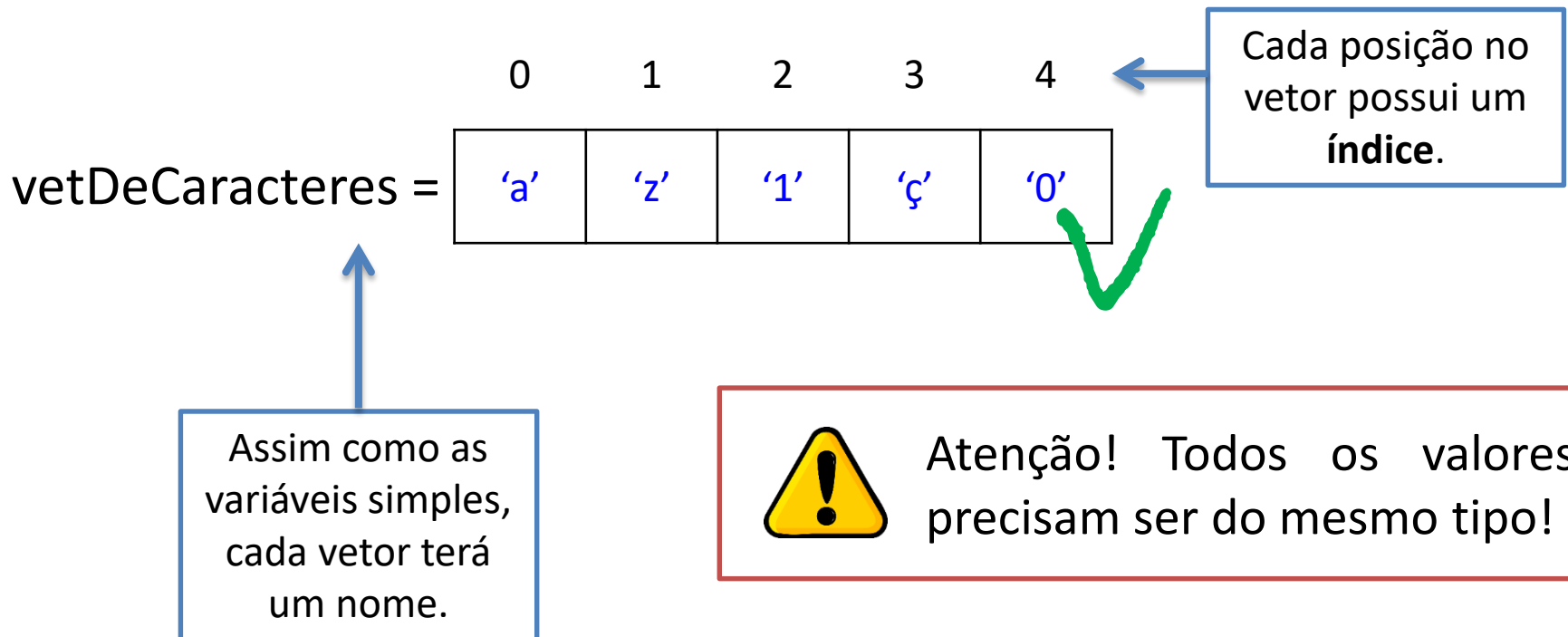


Atenção! Todos os valores precisam ser do mesmo tipo!

Vetores

- **Estrutura**

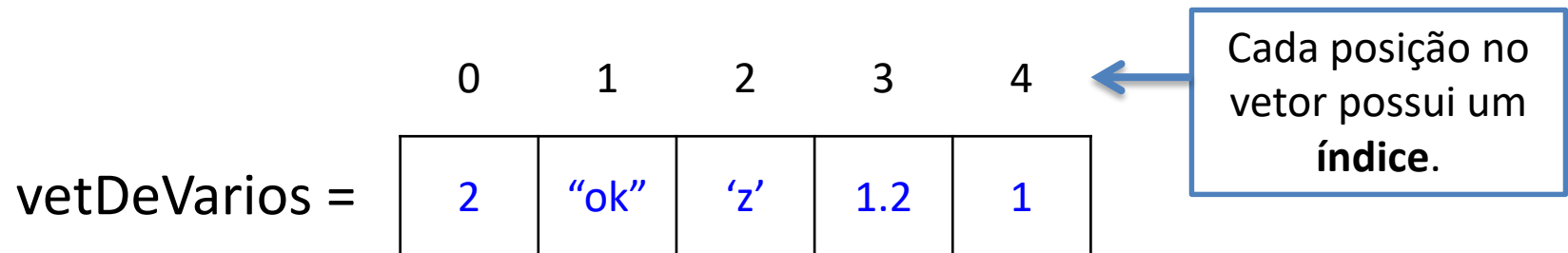
- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Estrutura**

- Um vetor é constituído por uma série de elementos em sequência.



Assim como as variáveis simples, cada vetor terá um nome.

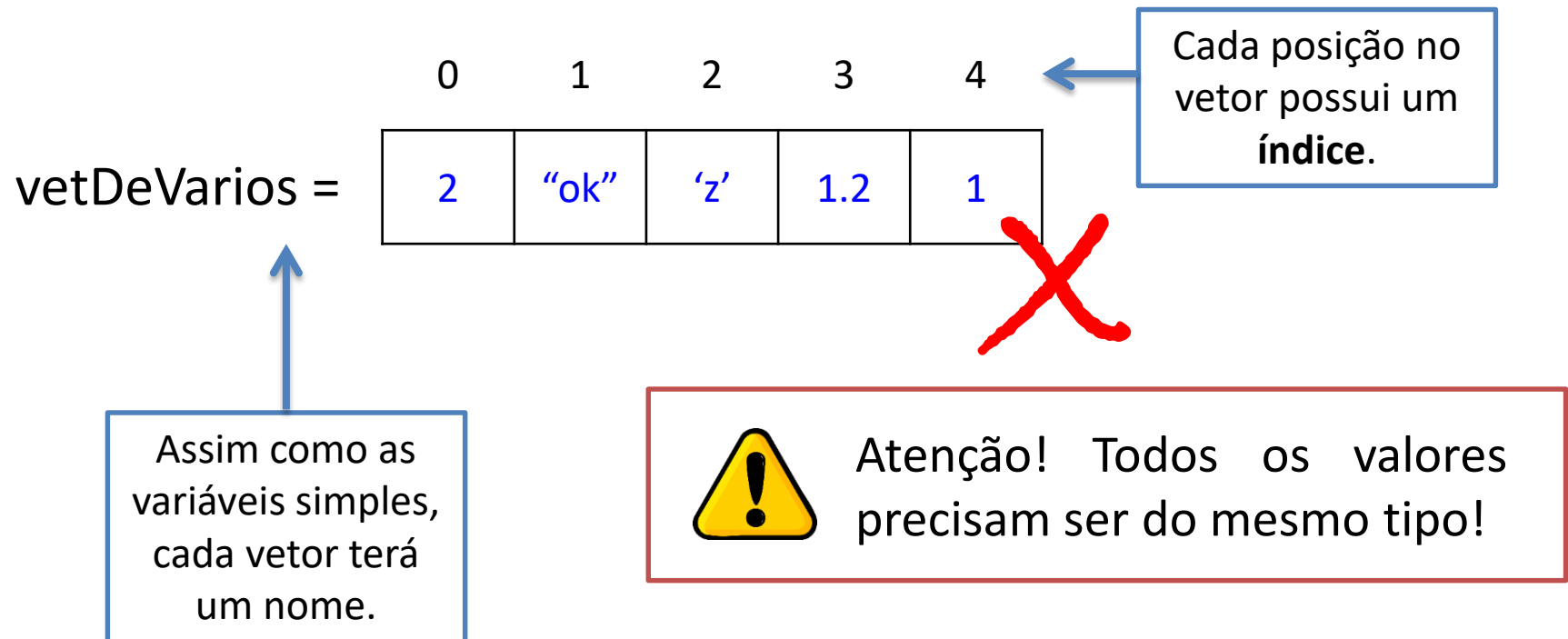


Atenção! Todos os valores precisam ser do mesmo tipo!

Vetores

- **Estrutura**

- Um vetor é constituído por uma série de elementos em sequência.



Vetores

- **Sintaxe**

- Para criar um vetor, é necessário fazer uma **declaração**.

`tipo identificador[tamanho]`

Vetores

- **Sintaxe**

- Para criar um vetor, é necessário fazer uma **declaração**.

tipo identificador[tamanho]

Um dos tipos possíveis:

caracter cadeia inteiro
real logico

Vetores

- **Sintaxe**

- Para criar um vetor, é necessário fazer uma **declaração**.

tipo identificador[tamanho]

Um dos tipos possíveis:

caracter cadeia inteiro
real logico

Qualquer identificador
seguindo as **3 restrições**.

Vetores

- **Sintaxe**

- Para criar um vetor, é necessário fazer uma **declaração**.

tipo **identificador** [tamanho]

Colchetes

Um dos tipos possíveis:

caracter **cadeia** **inteiro**
real **logico**

Qualquer identificador
seguindo as **3 restrições**.

Vetores

- **Sintaxe**

- Para criar um vetor, é necessário fazer uma **declaração**.

tipo **identificador** [tamanho]

Colchetes

Um dos tipos possíveis:

caracter **cadeia** **inteiro**
real **logico**

Qualquer identificador
seguindo as **3 restrições**.

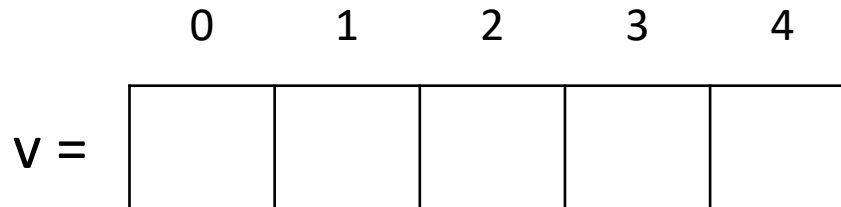
O **número máximo**
de valores que o
vetor consegue
guardar.

Vetores

- **Sintaxe**

- O vetor é uma estrutura de dados **estática**, ou seja, que não pode aumentar ou diminuir o seu número de elementos e este número deve ser especificado a priori.
- Por exemplo, este vetor foi criado com o **identificador v** e **tamanho 5**.

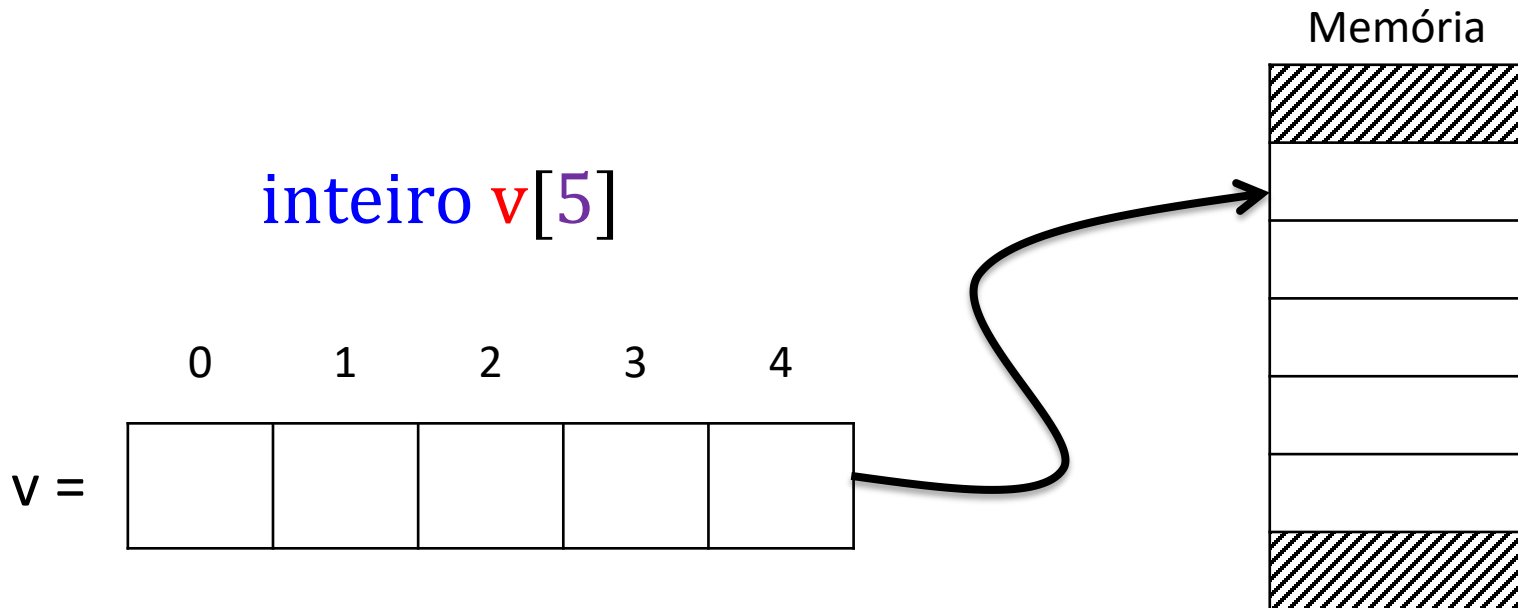
inteiro **v**[5]



Vetores

- **Exemplo**

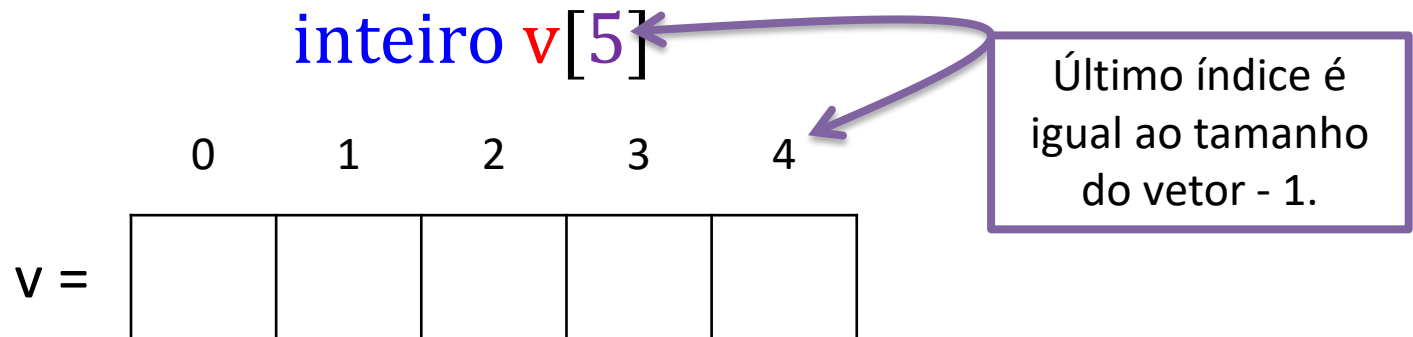
- Na memória do computador, **5 espaços** para elementos do tipo **inteiro** serão alocados **sequencialmente**.



Vetores

- **Exemplo**

- Cada elemento do vetor poderá ser **acessado diretamente** a partir de um **índice**.
- Em Portugal (assim como da maioria das outras linguagens) o **primeiro índice** do vetor é **0**.
- O **último índice** do vetor deverá ser a **dimensão** do vetor **decrecida** de uma unidade.



Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

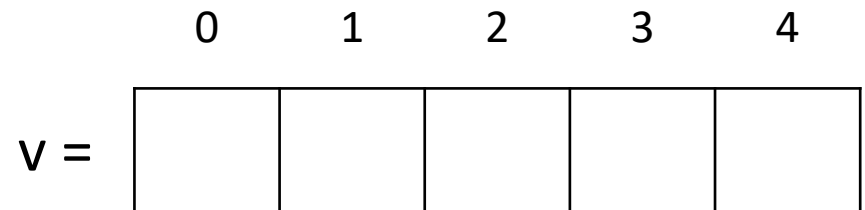
```
inteiro v[5]
```

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]
```

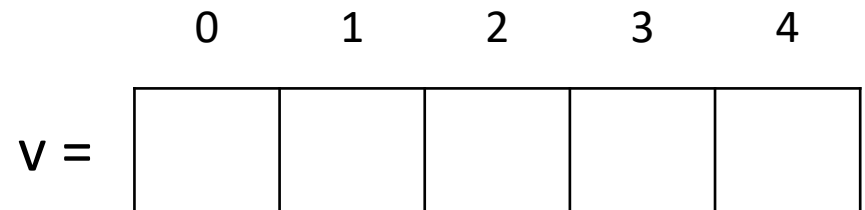


Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2
```

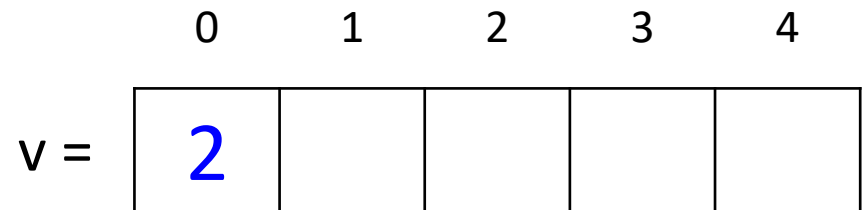


Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2
```



Vetores

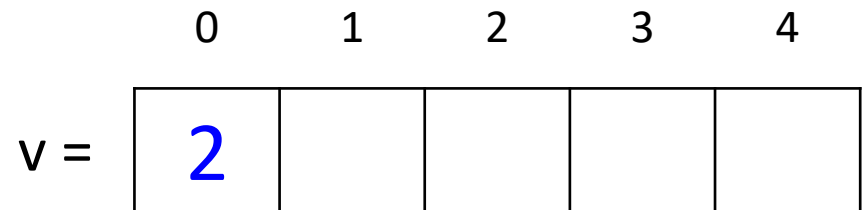
- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]
```

```
v[0] = 2
```

```
v[3] = 5
```



Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]
```

```
v[0] = 2
```

```
v[3] = 5
```

v =

0	1	2	3	4
2			5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]
```

v =

0	1	2	3	4
2			5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]
```

v =

0	1	2	3	4
2	5		5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]  
v[2] = v[0] + 1
```

v =

0	1	2	3	4
2	5		5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]  
v[2] = v[0] + 1
```

v =

0	1	2	3	4
2	5	3	5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]  
v[2] = v[0] + 1  
v[4] = v[2] + v[1]
```

v =

0	1	2	3	4
2	5	3	5	

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]  
v[2] = v[0] + 1  
v[4] = v[2] + v[1]
```

v =

0	1	2	3	4
2	5	3	5	8

Vetores

- **Exemplo**

- Para acessar os elementos, deve-se escrever o nome do vetor seguido de um índice entre colchetes.
- Por exemplo:

```
inteiro v[5]  
v[0] = 2  
v[3] = 5  
v[1] = v[3]  
v[2] = v[0] + 1  
v[4] = v[2] + v[1]
```

v =

0	1	2	3	4
2	5	3	5	8

Vetores

- **Sintaxe**

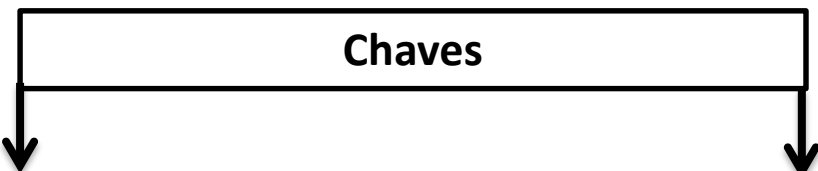
- Também é possível atribuir os valores iniciais dos elementos do vetor assim que ele for declarado.

`tipo identificador[N] = {valor1, valor2, ..., valorN}`

Vetores

- **Sintaxe**

- Também é possível atribuir os valores iniciais dos elementos do vetor assim que ele for declarado.



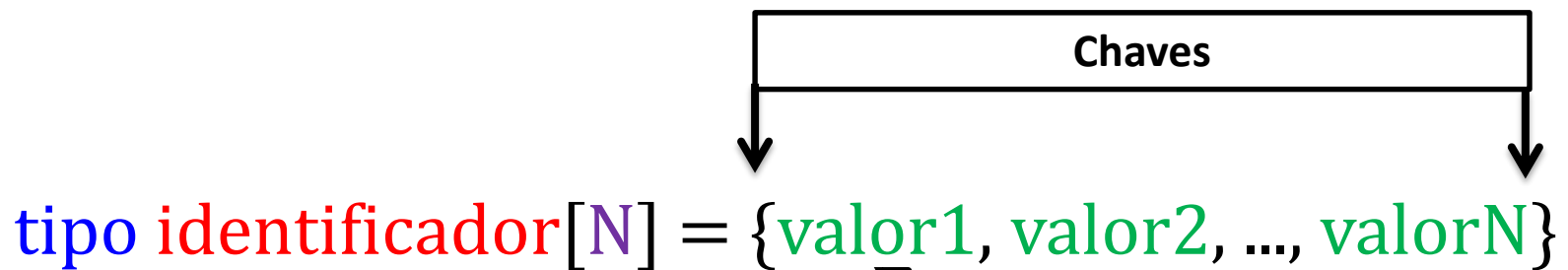
The diagram consists of a horizontal rectangle labeled "Chaves" (Keys). Two vertical lines extend downwards from the left and right sides of this rectangle, each ending in a downward-pointing arrow. These arrows point to the square brackets "[N]" and the closing curly brace "}" of the array declaration below.

`tipo identificador[N] = {valor1, valor2, ..., valorN}`

Vetores

- **Sintaxe**

- Também é possível atribuir os valores iniciais dos elementos do vetor assim que ele for declarado.


The diagram illustrates the syntax of a vector declaration: `tipo identificador[N] = {valor1, valor2, ..., valorN}`. A box labeled "Chaves" (Brackets) has two arrows pointing to the opening and closing curly braces of the initialization list. Three arrows point from a box below to the individual values `valor1`, `valor2`, and `valorN` within the list.



Atenção! Se o vetor não for inicializado, ele estará preenchido com um valor padrão para cada tipo.

Todos os **valores** necessários para preencher o vetor, um para cada posição.

Vetores

- **Exemplo**

- Por exemplo:

```
cadeia v[4] = {"Marco", "André", "Abud", "Kappel"}
```

Vetores

- **Exemplo**

- Por exemplo:

```
cadeia v[4] = {"Marco", "André", "Abud", "Kappel"}
```

	0	1	2	3
v =	"Marco"	"André"	"Abud"	"Kappel"

Vetores

- **Exemplo**

- Por exemplo:

```
cadeia v[4] = {"Marco", "André", "Abud", "Kappel"}
```

	0	1	2	3
v =	"Marco"	"André"	"Abud"	"Kappel"

```
escreva(v[0], " ", v[3])
```

Vetores

- **Exemplo**

- Por exemplo:

```
cadeia v[4] = {"Marco", "André", "Abud", "Kappel"}
```

	0	1	2	3
v =	"Marco"	"André"	"Abud"	"Kappel"

```
escreva(v[0], " ", v[3])
```



> _ Console Mensagens

Marco Kappel
Programa finalizado. Tempo de execução: 20 milissegundos

Vetores

- **Definição do tamanho**

- Quando se usa vetores, uma **boa prática** é definir no início do programa uma **constante** com o **tamanho** dos vetores.
- Com este valor **centralizado**, fica mais fácil criar **vários vetores** com o **mesmo tamanho** e, se necessário, alterar o tamanho de **todos** eles.

```
const inteiro MAX = 10  
  
inteiro vetor[MAX]
```

Vetores

- **Percorrer um vetor**

- Em muitos problemas, precisamos **percorrer** um vetor até encontrar um elemento de interesse.
- Por exemplo, suponha que temos o seguinte vetor de números inteiros:

	0	1	2	3	4
v =	2	5	3	5	8

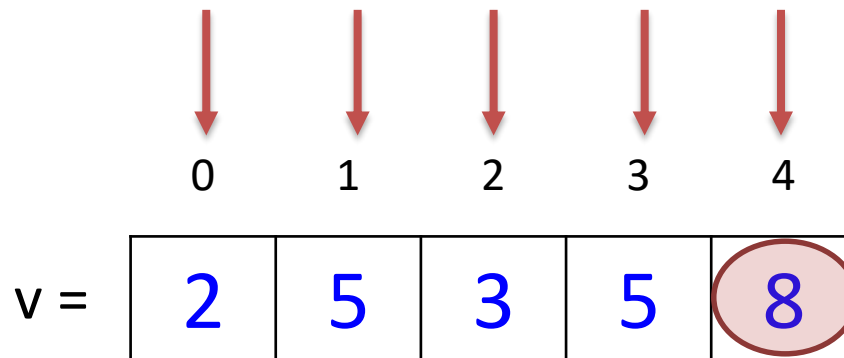
- Como faríamos para **verificar** se o número 8 está no vetor?

Vetores

- **Percorrer um vetor**

- Em muitos problemas, precisamos **percorrer** um vetor até encontrar um elemento de interesse.
- Por exemplo, suponha que temos o seguinte vetor de números inteiros:

Percorreríamos o vetor, elemento por elemento!



- Como faríamos para **verificar** se o número 8 está no vetor?

Vetores

- Percorrer um vetor
 - Ou seja, podemos usar uma **estrutura de repetição**!
 - **Exemplo**: Complete o código do programa abaixo:

```
const inteiro MAX = 5

inteiro v[MAX] = {2, 5, 3, 5, 8}

para(                ){
    se(                ){
        escreva("Encontrei o 8 no índice ", )
    }
}
```

Vetores

- **Exemplo**

- Dados o vetor e os comandos abaixo, qual será o conteúdo final do vetor vet?

vet =

7	4	1	5	8	2	3	6
---	---	---	---	---	---	---	---

```
inteiro i = 7

enquanto(i > 4){
    aux = vet[i]
    vet[i] = vet[7-i]
    vet[7-i] = aux
    i = i - 1
}
```

Vetores

- **Passar vetor para função**

- Podemos passar um **vetor** como **argumento** para uma **função**.
- Na chamada da função, deve-se fazer o **mesmo** que se fazia com uma variável, passando o **nome do vetor**.
- Na **declaração da função**, deve-se colocar **colchetes** na frente do identificador para mostrar que se trata de um **vetor**.
- Por exemplo, a seguinte função **recebe um vetor** e imprime seus elementos na tela:

`escreveVetor(v)`



```
funcao escreveVetor(inteiro v[]){  
    escreva("O vetor é: \n")  
    para(inteiro i = 0; i < MAX; i++){  
        escreva (v[i], " ")  
    }  
}
```

Vetores

- **Retornar vetor de função**
 - Não é possível **retornar** um vetor **diretamente** de uma função.
 - Mas podemos fazer isso **indiretamente**. Como?

Vetores

- **Retornar vetor de função**

- Não é possível **retornar** um vetor **diretamente** de uma função.
- Mas podemos fazer isso **indiretamente**. Como?
- Passagem de valor por **referência**!
- Basta colocar o **&** indicando que o vetor está sendo passado desta forma.
- Por exemplo, a função abaixo **preenche um vetor** com valores digitados pelo usuário:

```
funcao leVetor(inteiro &v[]){  
    para(inteiro i = 0; i < MAX; i++){  
        escreva("Digite o elemento de indice ",i,":")  
        leia(v[i])  
    }  
}
```

Vetores

- Retornar vetor de função

- Em Portugol, mesmo que você se esqueça de inserir o **&** ao lado do vetor, ele será **sempre** passado por referência.

```
funcao leVetor(inteiro &v[]){  
    para(inteiro i = 0; i < MAX; i++){  
        escreva("Digite o elemento de indice ",i,":")  
        leia(v[i])  
    }  
}
```

Passagem por
referência

```
funcao leVetor(inteiro v[]){  
    para(inteiro i = 0; i < MAX; i++){  
        escreva("Digite o elemento de indice ",i,":")  
        leia(v[i])  
    }  
}
```

- **Exemplo:**

- Implemente o seguinte programa, que cria um vetor, lê e imprime os seus valores:

```
programa{

    const inteiro MAX = 3

    funcao inicio(){
        inteiro v[MAX]
        leVetor(v)
        escreveVetor(v)
    }

    funcao escreveVetor(inteiro v[]){
        escreva("O vetor é: \n")
        para(inteiro i = 0; i < MAX; i++){
            escreva (v[i], " ")
        }
    }

    funcao leVetor(inteiro &v[]){
        para(inteiro i = 0; i < MAX; i++){
            escreva("Digite o elemento de indice ",i,":")
            leia(v[i])
        }
    }
}
```

• Exercícios

1. Faça um programa que leia dois vetores a e b contendo 20 elementos inteiros cada. Depois, o programa deve gerar e exibir um vetor c com o mesmo tamanho, cujos elementos sejam a soma dos respectivos elementos de a e b. Exemplo:

A =	23	37	30	...	45	35	B =	30	32	46	...	33	42	C =	53	69	76	...	88	77
	0	1	2		18	19		0	1	2		18	19		0	1	2		18	19

2 . Faça um programa que leia um vetor a contendo 20 elementos inteiros. Depois, o programa deve gerar e exibir um vetor b cujos elementos estão na ordem inversa de a. Exemplo:

A =	23	37	...	45	35
	0	1		18	19
B =	35	45	...	37	23
	0	1		18	19

• Exercícios

3. Faça um programa que leia dois vetores a e b contendo 25 elementos inteiros cada. Depois, o programa deve gerar e exibir um vetor c de 50 elementos, cujos elementos sejam a intercalação dos elementos de A e B.

Exemplo:

A =	23	37	30	...	38	55					
	0	1	2		23	24					
B =	30	32	46	...	43	49					
	0	1	2		23	24					
C =	23	30	37	32	30	46	...	38	43	55	49
	0	1	2	3	4	5		46	47	48	49

4. Faça um programa que:

- Leia um vetor de 10 números inteiros.
- Conte e imprima quantos pares existem no vetor.
- Conte e imprima quantos números no intervalo fechado de 1 a 10 aparecem no vetor.

Obs: Crie uma função para cada item.

• Exercícios

5. Faça uma função que recebe dois vetores de 10 elementos. Cada índice do vetor corresponde a uma pessoa. O primeiro vetor guarda as idades das 10 pessoas, enquanto o segundo vetor guarda o peso delas. A função deve retornar o peso médio das pessoas com mais de 30 anos.

6. Faça uma função que recebe um vetor e dois índices x e y. A função deve encontrar a posição dos elementos x e y do vetor.

7. Faça um programa que lê o preço de compra e o preço de venda de 100 mercadorias. O algoritmo deverá imprimir quantas mercadorias proporcionam:

- a) lucro menor do que 10%,
- b) lucro entre 10% e 20%, inclusive, e
- c) lucro maior que 20%.

• Exercícios

8. Faça um programa para corrigir provas de múltipla escolha com 10 questões. Cada questão vale 1 ponto. O primeiro conjunto de dados a ser lido será o gabarito da prova. Depois, serão lidos os números dos alunos e suas respectivas respostas. O número do aluno que provoca o término destas leituras será 0 (zero).

O programa deverá calcular e imprimir:

- Para cada aluno, o seu número e sua nota.
- O percentual de aprovação, sabendo que a nota mínima de aprovação é 7.
- A nota que teve a maior frequência absoluta, ou seja, que apareceu em maior número de vezes.

A estrutura de dados para este programa deve ser a seguinte:

GABARITO

--	--	--	--	--	--	--	--	--	--

RESPOSTAS

--	--	--	--	--	--	--	--	--	--

FREQUENCIA

--	--	--	--	--	--	--	--	--	--	--

NUMERO

--

NOTA

--

APROVADOS

--

TOTAL

--

MAIOR

--

PORCENTAGEM

--

- **Exercícios**

9. Um sistema de controle de estoque armazena o código, a descrição, a quantidade em estoque e o preço unitário das mercadorias. Faça um programa que exiba um menu com as seguintes opções:

MENU

- 1 - Cadastrar mercadoria
- 2 - Consultar mercadoria
- 3 - Valor total em mercadorias
- 4 - Sair

Inicialmente, não há nenhuma mercadoria cadastrada. A primeira opção permitirá que o usuário cadastre uma nova mercadoria, informando todos os dados citados anteriormente. A segunda opção permitirá que o usuário consulte a descrição da mercadoria, informando seu código. A terceira opção permitirá a consulta do valor total do estoque, levando em conta todas as mercadorias cadastradas. A quarta opção permitirá o término do programa. O sistema deve aceitar, no máximo, 100 mercadorias.

Vetores

FIM