# Vetores (Arrays)

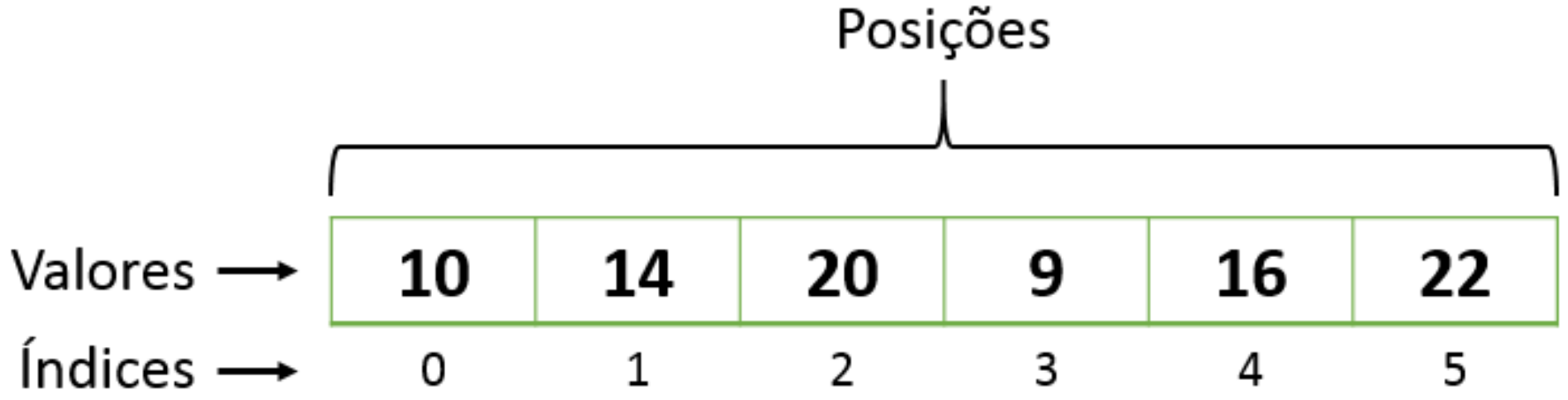
Prof. Thiago Felski Pereira, MSc.

# Definição

- Imagine que fosse necessário criar um programa para armazenar 2000 salários dos funcionários de uma empresa.
- Caso só se quisesse somar os salários poderia se criar um laço de repetição, ler 2000 vezes a mesma variável, e ir acumulando os valores.
- Mas, e se fosse necessário acessar esses salários novamente para outras funções?
  - Os valores estariam perdidos. Então, neste caso, seria necessário criar 2000 variáveis.
- Criar 2000 variáveis se torna inviável. Para resolver isso usam-se arrays, que podem ser tanto vetores quanto matrizes.

# Definição

- Mas, o que é um vetor?
  - Uma sequência de vários valores de mesmo tipo, armazenados sequencialmente na memória, e fazendo uso de um mesmo nome de variável para acessar esses valores.
  - Vetores são variáveis que servem para guardar vários valores do mesmo tipo, de forma uniforme, na memória.






UNIVALI

# Definição

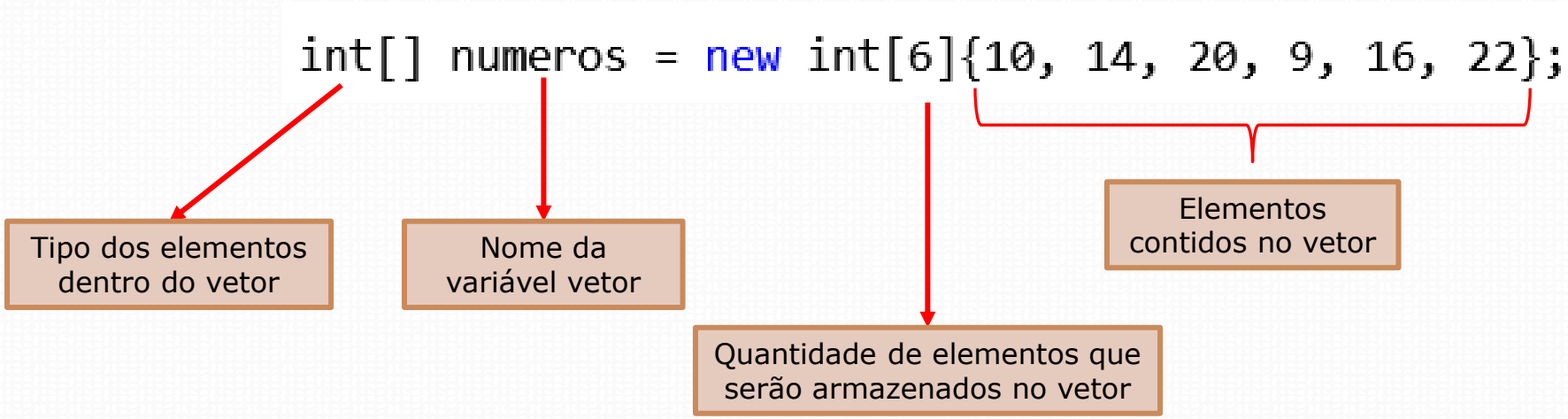
- E como se declara um vetor?
  - A diferença entre a declaração de um vetor e de uma variável normal é o uso de [].
  - Primeiro, coloca-se o tipo de dados que serão armazenados no vetor, ou seja, se o vetor conterá valores inteiros, reais, caracteres, …
  - Logo após [] indicando que a variável será um vetor.
  - Então, se dá um nome para o vetor (assim como as variáveis).
    - Vale lembrar que as regras para nomear uma variável também são válidas para um vetor.
  - Finalmente, atribuem-se valores a esse vetor.

- Exemplo:
  - Criação de um vetor de inteiros com 6 posições.
  - *Vetor do slide anterior.*

```
int[] numeros = new int[6]{10, 14, 20, 9, 16, 22};
```

# Definição

```
int[] numeros = new int[6]{10, 14, 20, 9, 16, 22};
```




- Também é possível criar um vetor e deixar para colocar os valores nele depois.

```
int[] numeros = new int[6];
```



UNIVALI

# Definição

- Erros comuns ao se declarar um vetor em C#

```
// falta definir o tamanho
int[] evenNums = new int[];

// o número de elementos deve ser igual ao tamanho especificado
int[] evenNums = new int[5] { 2, 4 };

// A array não foi iniciada corretamente.
var evenNums = { 2, 4, 6, 8, 10};
```

# Manipulação de Vetores

- Considere que a variável vet possui 4 espaços de memória onde podem ser guardados 4 números inteiros.

- Se fosse uma variável simples, bastaria chamar pelo nome da mesma.
  - Por exemplo, para imprimir o conteúdo da variável `numero` basta utilizar a linha a seguir: `Console.WriteLine(numero);`

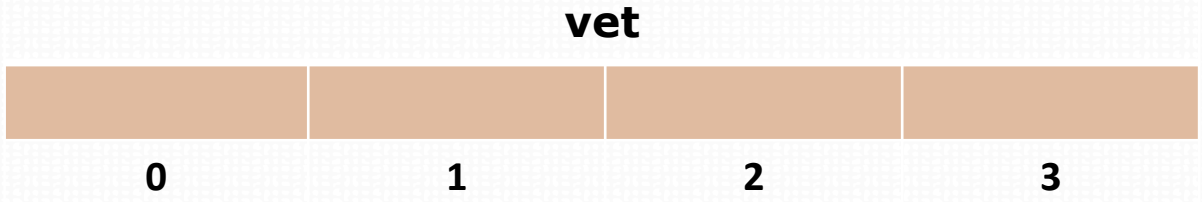
- Mas o vetor possui mais que uma posição. Como acessar um determinado elemento?




UNIVALI

# Manipulação de Vetores

- O vetor trabalha com índices, ou posições, que iniciam de 0 (sempre)




- Para inserir o valor 76 na posição 2 seria necessário o seguinte código:
  - `vet[2] = 76;`
- Ou seja, para atribuir um valor a uma posição do vetor é como uma variável, com o acréscimo do índice.
- É como um carteiro. Para entregar a carta ele precisa saber o nome da rua e o número da residência.
- O nome da rua não deixa de ser o nome do vetor, e o número da residência é o índice do vetor.

# Manipulação de Vetores

- **LEMBRE-SE**: o limite do vetor é sempre seu tamanho menos 1.

- Usando o exemplo:
  - Um vetor de tamanho 4, posição máxima é 3 (pois 4-1=3).
  - Então, ao atribuir um valor a posição 4 ocorrerá um erro.
  - Resumidamente, jamais se poderia fazer `vet[4]=200;` pois a última posição do vetor é a posição 3.

# Manipulação de Vetores

- **EXEMPLO**: Faça um programa que preencha um vetor de 10 posições e exiba o mesmo na tela.

# Manipulação de Vetores

- **EXEMPLO 1**: Faça um programa que preencha um vetor de 10 posições e exiba o mesmo na tela.

```
1  using System;
2  public class Program {
3      public static void Main() {
4
5          int[] vetor = new int[10];
6
7          for (int i = 0; i < 10; i++) {
8              Console.WriteLine("vetor[" + i + "] = ");
9              vetor[i] = int.Parse( Console.ReadLine() );
10         }
11     }
12 }
```

# Manipulação de Vetores

- Ao ler uma variável, acessamos somente uma posição. No caso do vetor é mais que uma posição acessada pela mesma variável.

- Imagine um vetor de 1000 posições, sendo lido posição a posição:
  - `vetor[0] = int.Parse( Console.ReadLine() );`
  - `vetor[1] = int.Parse( Console.ReadLine() );`
  - `...`
  - `vetor[999] = int.Parse( Console.ReadLine() );`
- Dessa forma, fica melhor se utilizar um laço de repetição para acessar as posições.
- Seguem outros exemplos.

# Manipulação de Vetores

- **EXEMPLO 2**: Faça um programa que preencha um vetor de 20 posições e exiba todos os valores pares e suas respectivas posições.

# Manipulação de Vetores

- **EXEMPLO 2**: Faça um programa que preencha um vetor de 20 posições e exiba todos os valores pares e suas respectivas posições.

```
1  using System;
2  public class Program {
3      public static void Main() {
4          int[] vetor = new int[20];
5          for (int i = 0; i < 20; i++) { //Inserindo valores no vetor
6              Console.WriteLine("vetor[" + i + "] = ");
7              vetor[i] = int.Parse( Console.ReadLine() );
8          }
9          for (int i = 0; i < 20; i++) { //Buscando e imprimindo os pares
10             if (vetor[i] % 2 == 0) {
11                 Console.WriteLine("vetor[" + i + "] = " + vetor[i]);
12             }
13         }
14     }
15 }
```

# Manipulação de Vetores

- Existe, normalmente, confusão ao que se refere a acessar o elemento dentro do vetor e o índice.

- Toda vez que é necessário acessar o elemento dentro do vetor é necessário a composição nome do vetor e posição a ser acessada.
  - seja a posição fixa ou representada por um índice.

- No exemplo anterior era necessário verificar os valores, ou seja, o que foi inserido pelo usuário no vetor.
  - Então, para acessar é utilizado `vetor[i]` (nome do vetor e posição representada por índice).
  - Mas ao se mostrar somente a posição, é a variável `i` (neste caso) que é exibida, pois ela representa a posição.

# Manipulação de Vetores

- **EXEMPLO 2**: Elabore um algoritmo que preencha um vetor de 15 posições e o imprima de forma invertida (de trás para frente).

# Manipulação de Vetores

- **EXEMPLO 2**: Elabore um algoritmo que preencha um vetor de 15 posições e o imprima de forma invertida (de trás para frente).

```
1  using System;
2  public class Program {
3      public static void Main() {
4          int[] vetor = new int[15];
5          for (int i = 0; i < 15; i++) { //Inserindo valores no vetor
6              Console.WriteLine("vetor[" + i + "] = ");
7              vetor[i] = int.Parse( Console.ReadLine() );
8          }
9          for (int i = 14; i >= 0; i--) { //Imprimindo invertido
10             Console.WriteLine(vetor[i]);
11         }
12     }
13 }
```

# Manipulação de Vetores

- O vetor foi preenchido de forma normal.
- A grande diferença é que o vetor precisa ser impresso de forma invertida, ou seja, iniciar da última posição. Foi criado um laço que iniciou da posição 14, ou seja, a última posição e foi decrementando até a posição 0.

- Este é um erro efetuado constantemente. Quando se precisa acessar a última posição se utilizar o tamanho do vetor. Lembre-se: a quantidade total de elementos é usado na declaração do vetor.
  - Para acessar a última posição do vetor, como o índice inicia de 0, é sempre tamanho – 1, ou seja, se tiver 100 posições declaradas a última posição é 99.

# Obrigado pela atenção

contato: felski@univali.br