IFRN

PROGRAMAÇÃO ORIENTADA A OBJETOS EM C#

Vetores e Matrizes

Prof. Gilbert Azevedo

Objetivos

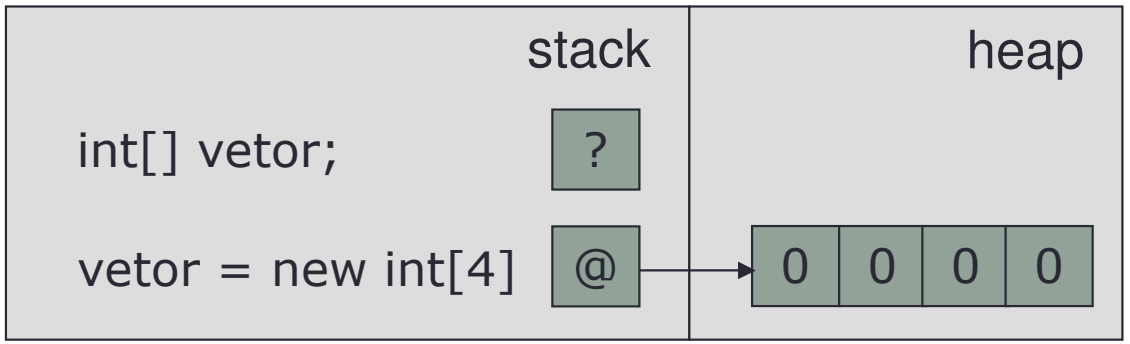
- Conhecer o tipo de dados Vetor e sua utilização no C#
  - Conceito
  - Declaração, instanciação, inicialização e acesso
  - Propriedades e métodos da classe Array
  - Iteração em vetores
- Conhecer o tipo de dados Matriz e sua utilização no C#
  - Declaração, instanciação, inicialização e acesso
  - Escrita formatada de matrizes

Vetores

- Tipo de dados que representa uma coleção de elementos acessados por um índice inteiro
- O índice indica posição de um elemento no vetor e inicia em zero
- Vetores são tipos por referência e devem ser instanciados (new)
- Independente do tipo dos elementos, descende de System.Array
  - Propriedades e métodos da classe Array podem ser usados em quaisquer vetores: tamanho, copia, inversão, ordenação, ...

Declarando e Instanciando Vetores

- Declaração de Vetores – Referências
  - É necessário definir o tipo do elemento, seguido de um par de chaves e de uma variável para referenciar o vetor
  - `int[] vetor; // Referência`
- Alocação de Vetores – Instâncias
  - O operador *new* é usado para alocar o vetor na memória
  - `int[] vetor = new int[4]; // Vetor`



Iniciando os Elementos do Vetor

- Os elementos podem ser iniciados na criação do vetor
  - `int[] vetor = new int[4] { 1, 2, 3, 4 };`
- Sintaxe sem o operador new
  - `int[] vetor = { 1, 2, 3, 4 };`
- O número de elementos do vetor pode ser uma variável mas não pode ser alterado após sua criação
  - `int n = 4;`
  - `int[] vetor = new int[n];`

Acessando os Elementos do Vetor

- Os elementos são acessados com o operador de indexação [ ]
  - Uma exceção IndexOutOfRangeException é lançada se um índice inválido é usado: menor que zero, maior ou igual ao nº de elementos
- Recuperando o valor de um elemento
  - `int[] vetor = new int[4] { 1, 2, 3, 4 };`
  - `int total = vetor[0] + vetor[1] + vetor[2] + vetor[3];`
  - `Console.WriteLine(vetor[0]);`
- Atribuindo valor a um elemento
  - `vetor[0] = 0;`
  - `vetor[4] = 0;  // IndexOutOfRangeException`

Propriedades da Classe Array

- A propriedade Length retorna o número de elementos do vetor
  - `int[] vetor = new int[4] { 1, 2, 3, 4 };`
  - `Console.WriteLine(vetor.Length); // 4`
- Rank retorna o número de dimensões do vetor
  - `Console.WriteLine(vetor.Rank);  // 1`
- Length e Rank são propriedades somente de leitura
  - Número de elementos e dimensão são determinados quando o vetor é instanciado

Iterando em um Vetor

- Iteração com repetição for, while, do-while usando o índice
  - `for (int i = 0; i < vetor.Length; i++)`
  - `  Console.WriteLine(vetor[i]);  // i é o índice`
- Iteração com foreach – Acessa o elemento sem usar o índice
  - `foreach (int i in vetor)`
  - `  Console.WriteLine(i);  // i é o elemento`
- Iteração com IEnumerator – Enumerador é um padrão da POO
  - `IEnumerator x = vetor.GetEnumerator();`
  - `while (x.MoveNext()) Console.WriteLine(x.Current);`

Métodos da Classe Array

- O operador de atribuição realiza apenas uma cópia da referencia
  - `int[] v = { 1, 2, 3, 4 };`
  - `int[] x = v;`
  - x e v são referências para um mesmo objeto int[]
- Método estático Copy: copia um número determinado de elementos de um vetor para outro
  - `int[] v = new int[4] { 1, 2, 3, 4 };`
  - `int[] c = new int[4];`
  - `Array.Copy(v, c, 2);  // c = { 1, 2, 0, 0 }`

Métodos da Classe Array

- Método CopyTo: copia todos elementos de um vetor para outro, começando no índice informado no vetor de destino
  - `int[] v = new int[4] { 1, 2, 3, 4 };`
  - `int[] c = new int[4];`
  - `v.CopyTo(c, 0);  // c = { 1, 2, 3, 4 }`
  - O vetor de destino deve comportar todos os elementos do vetor origem
- Método Clone: realiza um cópia do vetor sem precisar instanciar o vetor de destino. Requer um *type-cast* na operação.
  - `int[] c;`
  - `c = (int[]) v.Clone();  // c = { 1, 2, 3, 4 }`

Métodos da Classe Array

- BinarySearch: retorna o índice de um elemento no vetor ordenado
- Clear: reinicia o valor dos elementos
- Contains: verifica se o vetor contém um determinado elemento
- GetEnumerator: retorna um enumerador para o vetor
- GetLength: retorna o número de elementos em uma dimensão
- IndexOf: retorna o índice de um elemento no vetor
- Resize: altera o número de elementos do vetor
- Reverse: Inverte a ordem dos elementos
- Sort: ordena os elementos do vetor

Exemplo

- Exemplo: Ler um vetor de 10 inteiros e mostrar na ordem inversa

```
public static void Main (string[] args) {
  int[] x = new int[10];
  Console.WriteLine("Digite 10 valores inteiros");
  for (int i = 0; i < 10; i++)
    x[i] = int.Parse(Console.ReadLine());
  Console.WriteLine("Ordem inversa");
  for (int i = 9; i >= 0; i--)
    Console.WriteLine(x[i]);
}
```

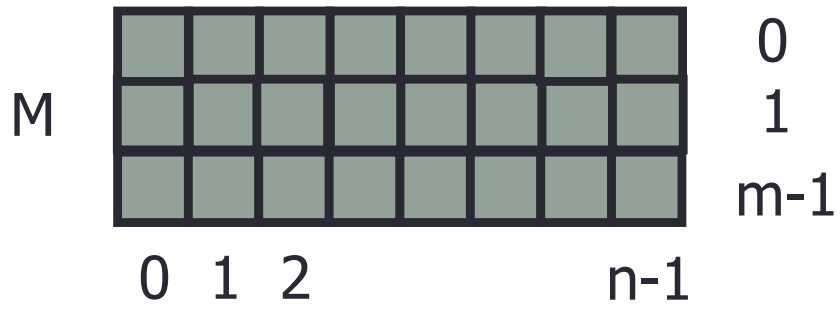
Exemplo

- Exemplo: Ler um vetor de 10 inteiros e mostrar na ordem inversa

```
public static void Main (string[] args) {
  int[] x = new int[10];
  Console.WriteLine("Digite 10 valores inteiros");
  for (int i = 0; i < 10; i++)
    x[i] = int.Parse(Console.ReadLine());
  Array.Reverse(x);
  Console.WriteLine("Ordem inversa");
  foreach (int i in x) Console.WriteLine(i);
}
```

Matrizes

- Tipo de dados que representa uma coleção bidimensional de valores composta por elementos de um mesmo tipo
- Uma variável do tipo matriz armazena diversos valores que são referenciados por dois índices: o primeiro índice representa a linha do elemento e o segundo, a coluna. Ambos índices iniciam em zero.



Declarando e Instanciando Matrizes

- Declaração de Matrizes – Referências
  - É necessário definir o tipo do elemento, seguido de um par de chaves com uma vírgula e de uma variável para referenciar a matriz
  - Ex: Declaração de uma matriz de inteiros
  - `int[,] matriz;`
- Alocação de Matrizes – Instâncias
  - O operador *new* é usado para alocar a matriz
  - Ex: Declarando e alocando uma matriz 3x4 de inteiros
  - `int[,] matriz = new int[3,4];`

Acessando os Elementos da Matriz

- Os elementos podem ser iniciados na criação da matriz e são acessados com os índices de linha e coluna
- Matriz 3x4 de elementos inteiros
  - `int[,] x = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };`
  - `int[,] x = new int[3,4] { {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };`
- Atribui 10 ao elemento da 1ª linha, 2ª coluna
  - `x[0,1] = 10;`
- Soma os elementos da 1ª linha da matriz x
  - `int i = x[0,0] + x[0,1] + x[0,2] + x[0,3];`

Escrevendo uma Matriz na Saída

- Escrita formatada da matriz

```
public static void Main (string[] args) {
  int[,] x = new int[3,4] { { 1,2,3,4 }, { 5,6,7,8 },
    { 9,10,11,12 } };
  Console.WriteLine("Matriz");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 4; j++)
      Console.Write("{0,5}", x[i, j]);  // usa 5 caracteres
    Console.WriteLine();                // por elemento
  }
}
```

Referências

- Microsoft Visual C# 2010 – Passo a passo, John Sharp, Bookman, 2010
- Matrizes (Guia de Programação em C#)
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/arrays/