UNIDADE 3 – ARRAY DE OBJETOS

Nessa unidade veremos a funcionalidades dos Arrays na programação.

SUMÁRIO

1 ARRAYS

Suponha que o sistema do banco tenha que gerar listas com os números das contas de uma agência. Poderíamos declarar uma variável para cada número.

```
1  int numero1;
2  int numero2;
3  int numero3;
```

Tabela 1: Uma variável para cada número de conta

Contudo, não seria uma abordagem prática, pois uma agência pode ter uma quantidade muito grande de contas. Além disso, novas contas podem ser abertas todos os dias. Isso implicaria em alterações constantes no código fonte.

Quando desejamos armazenar uma grande quantidade de valores de um determinado tipo, podemos utilizar arrays. Um array é um objeto que pode armazenar muitos valores de um determinado tipo.

Podemos imaginar um array como sendo um armário com um determinado número de gavetas. E cada gaveta possui um rótulo com um número de identificação.

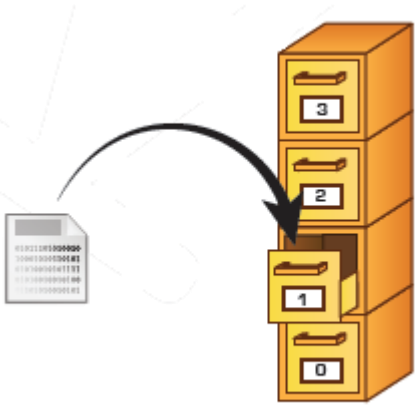

Figura 1: Analogia de array.

1.1 CRIANDO UM ARRAY

Em C#, os arrays são criados através do comando new.

```
1  int[] numeros = new int[100];
```

Tabela 2: Criando um array com capacidade para 100 valores do tipo int

A variável números armazena a referência de um array criado na memória do computador através do comando new. Na memória, o espaço ocupado por esse array está dividido em 100 "pedaços" iguais numerados de 0 até 99. Cada "pedaço" pode armazenar um valor do tipo int.

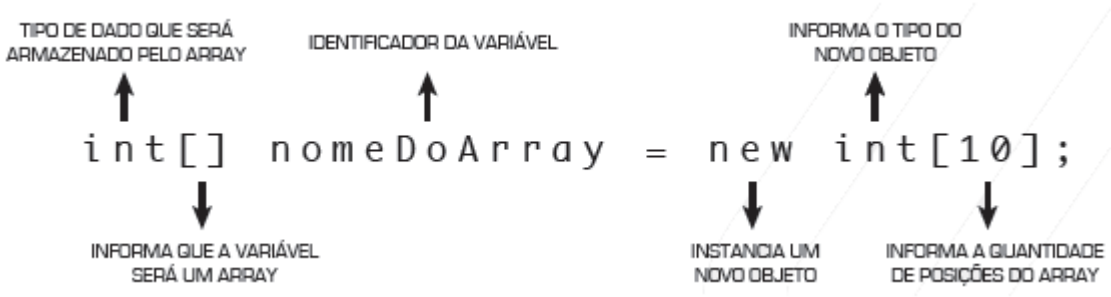

Figura 2: Criando um array.

1.2 MODIFICANDO O CONTEÚDO DE UM ARRAY

Para modificar o conteúdo de um array, devemos escolher uma ou mais posições que devem ser alteradas e utilizar a sintaxe abaixo:

```
1  int[] numeros = new int[100];
2  numeros[0] = 136;
3  numeros[99] = 17;
```

Tabela 3: Modificando o conteúdo das posições 0 e 99

Importante

Quando um array é criado com o comando new, todas as posições são inicializadas com os valores padrão (números são inicializados com 0, booleanos com false e referências com null).

Também podemos definir os valores de cada posição de um array no momento da sua criação utilizando as sintaxes abaixo:

```
1  int[] numeros = new int[2] { 100, 87 };
```

Tabela 4: Inicializando o conteúdo de um array

```
1  int[] numeros = new int[] { 100, 87 };
```

Tabela 5: Inicializando o conteúdo de um array

```
1  int[] numeros = { 100, 87 };
```

Tabela 6: Inicializando o conteúdo de um array

1.3 ACESSANDO O CONTEÚDO DE UM ARRAY

Para acessar o conteúdo de um array, devemos escolher uma ou mais posições e utilizar a sintaxe abaixo:

```
1  int[] numeros = { 100, 87 };
2  Console.WriteLine(numeros[0]);
3  Console.WriteLine(numeros[1]);
```

Tabela 7: Acessando o conteúdo das posições 0 e 1

Importante

Acessar posições fora do intervalo de índices de um array gera erro de execução. Mais especificamente, em C#, ocorrerá a exception IndexOutOfRangeException.

1.4 PERCORRENDO UM ARRAY

Quando trabalhamos com um array, uma das tarefas mais comuns é acessarmos todas ou algumas de suas posições sistematicamente. Geralmente, fazemos isso para resgatar todos ou alguns dos valores armazenados e realizar algum processamento sobre tais informações.

Para percorrermos um array, utilizaremos a instrução de repetição for. Podemos utilizar a instrução while também. Porém, logo perceberemos que a sintaxe da instrução for, em geral, é mais apropriada quando estamos trabalhando com arrays.

```
1  int[] numeros = new int[100];
2  for (int i = 0; i < 100; i++)
3  {
4      numeros[i] = i;
5  }
```

Tabela 8: Percorrendo um array

Para percorrer um array, é necessário saber a quantidade de posições do mesmo. Essa quantidade é definida quando o array é criado através do comando new. Nem sempre essa informação está explícita no código. Por exemplo, considere um método que imprima na saída padrão os valores armazenados em um array. Provavelmente, esse método receberá como parâmetro um array e a quantidade de posições desse array não estará explícita no código fonte.

```
1  void ImprimeArray(int[] numeros)
2  {
3      // implementação
4  }
```

Tabela 9: Método que deve imprimir o conteúdo de um array de int

Podemos recuperar a quantidade de posições de um array acessando o seu atributo Length.

```
1  void ImprimeArray(int[] numeros)
2  {
3      for (int i = 0; i < numeros.Length; i++)
4      {
5          Console.WriteLine(numeros[i]);
6      }
7  }
```

Tabela 10: Método que deve imprimir o conteúdo de um array de int

1.5 FOREACH

Para acessar todos os elementos de um array, é possível aplicar o comando foreach.

```
1  void ImprimeArray(int[] numeros)
2  {
3      foreach (var numero in numeros)
4      {
5          Console.WriteLine(numeros);
6      }
7  }
```

Tabela 11: Percorrendo um array com foreach

1.6 Operações

Nas bibliotecas da plataforma .NET, existem métodos que realizam algumas tarefas úteis relacionadas a arrays. Veremos esses métodos a seguir.

1.6.1 Ordenando um Array

Considere um array de string criado para armazenar nomes de pessoas. Podemos ordenar esses nomes através do método Array.Sort().

```
1  string[] nomes = new string[] { "Thiago Sartor ", "Alexandre Rech", "Tiburcio" };
2  Array.Sort(nomes);
3
4  foreach (string nome in nomes)
5  {
6      Console.WriteLine(nome);
7  }
```

Tabela 12: Ordenando um array

Analogamente, também podemos ordenar números.

1.7 Duplicando um Array

Para copiar o conteúdo de um array para outro com maior capacidade, podemos utilizar o método CopyTo().

```
1  string[] nomes = new string[] { "Thiago", "Alexandre", "Tiburcio" };
2  string[] nomesDuplicados = new string[3];
3  nomes.CopyTo(nomesDuplicados, 0);
```

Tabela 13: Duplicando

1.8 Exercícios de Fixação

1) Crie um projeto Unidade 3, dentro do projeto crie a pasta Arrays/ExercicioFixacao.

2) Crie um programa que imprima na tela os valores de um array já inicializados.

```
1  class ImprimeValores
2  {
3      0 references
4      static void Main()
5      {
6          int[] numeros = new int[] { 100, 87, 5, 78, 89, 45 };
7          foreach (var numero in numeros)
8          {
9              Console.WriteLine(numero);
10         }
11     }
12 }
```

Tabela 14: ImprimeValores.cs

Compile o ImprimeValores.

3) Faça um programa que ordene o array de strings.

```
1  class OrdenaValores
2  {
3      0 references
4      static void Main()
5      {
6          string[] nomes = new string[] { "Thiago","Yago","Matheus","Sabrina","Camila"};
7          Array.Sort(nomes);
8          foreach (string nome in nomes)
9          {
10             Console.WriteLine(nome);
11         }
12     }
13 }
```

Tabela 15:OrdenaValroes.cs

Compile o OrdenaValores.

1.9 EXERCÍCIOS COMPLEMENTARES

1) Faça um programa que calcule a média dos elementos recebidos do teclado. Dica: para converter strings para double utilize o método ToDouble().

```
1  Console.WriteLine("Digite um número:"); // Texto (string)
2  double numero = Convert.ToDouble(Console.ReadLine();) //Conversão
3  double numero2 = double.Parse(Console.ReadLine();) //Conversão
```

Tabela 16: ToDouble()

2) Faça a conversão para outros tipos, int, bool entre outros.

3) Crie um programa que encontre o maior número entre 15 valores digitados.

Bons Estudos!