表3.2 数组的重点

|  |  |
| --- | --- |
| **说明** | **示例** |
| **声明：**  注意数据类型后的方括号。  声明多维数组要使用逗号；逗号数量+1 = 维数。 | // 1.  string[] languages; // 一维  int[,] cells; // 二维 |
| **赋值：**  new关键字可以省略，但只能是在声明的时候。  如果未在声明的同时赋值，以后实例化数组时必须使用new关键字。  可以不指定数据类型（从C# 3.0开始）。但是，没有数据类型，就不能指定数组大小。  无论数组的大小还是值，都不需要是字面值；它们可以在运行时确定。  如果同时指定了值和数组大小，那么这两者必须保持一致。数组可以在没有值的情况下赋给变量，但数组中每一项的值都会被初始化为其默认值。  如果没有指定值，那么必须同时指定数据类型和数组大小。 | // 2.  string[] languages = {  "C#", "COBOL", "Java",  "C++", "TypeScript", "Pascal",  "Python", "Lisp", "JavaScript"};  languages = new string[]{  "C#", "COBOL", "Java",  "C++", "TypeScript", "Pascal",  "Python", "Lisp", "JavaScript" };  // 多维数组赋值和初始化  int[,] cells = new[,]  {  {1, 0, 2},  {1, 2, 0},  {1, 2, 1}  };  languages = new string[9]; |
| **正向访问数组**  数组基于零，所以第一个元素的索引是0。  用方括号存储和获取数组数据。 | // 3.  string[] languages = new[]{  "C#", "COBOL", "Java",  "C++", "TypeScript", "Visual Basic",  "Python", "Lisp", "JavaScript"};  // 从languages数组获取第5个元素(TypeScript)  string language = languages[4];  // 输出“TypeScript”  Console.WriteLine(language);  // 获取数组倒数第3个元素(Python)  language = languages[^3];  // 输出“Python”  Console.WriteLine(language); |
| **逆向访问数组**  从C# 8.0开始，还可以从数组末尾索引。例如，^1对应数组最后一个元素，^3对应数组倒数第三个元素。 |  |
| **范围**  C# 8.0 允许使用范围操作符（..）来标识并提取由数组元素构成的一个数组，该操作符标识的范围从起始项开始，到结束项为止（但不包括结束项）。 | string[] languages = new[]{  "C#", "COBOL", "Java",  "C++", "TypeScript", "Visual Basic",  "Python", "Lisp", "JavaScript"};  Console.WriteLine($@"^3..^0: {  // Python, Lisp, JavaScript  string.Join(", ", languages[^3..^0])}");  Console.WriteLine($@"^3..: {  // Python, Lisp, JavaScript  string.Join(", ", languages[^3..])}");  Console.WriteLine($@" 3..^3: {  // C++, TypeScript, Visual Basic  string.Join(", ", languages[3..^3])}");  Console.WriteLine($@" ..^6: {  // C#, COBOL, Java  string.Join(", ", languages[..^6])}"); |