**Strings**

* In C#, strings are objects. Thus, **string** is a reference type.
* Although **string** is a built-in data type in C#, a discussion of **string** needed to wait until classes and objects had been introduced.

**Constructing Strings:**

* The easiest way to construct a **string** is to use a string literal.
* For example, here **str** is a **string** reference variable that is assigned a reference to a string literal:

string str = "C# strings are powerful.";

In this case, **str** is initialized to the character sequence “C# strings are

powerful.”

* You can also create a **string** from a **char** array. For example:

char[] charray = {'t', 'e', 's', 't'};

string str = new string(charray);

* Once you have created a **string** object, you can use it nearly anywhere that a quoted string is allowed. For example, you can use a **string** object as an argument to **WriteLine( )**, as shown in this example:

// Introduce string.

**using System;**

**class** StringDemo

{

**public static void Main()**

{

char[] charray = {'A', ' ', 's', 't', 'r', 'i', 'n', 'g', '.' };

string str1 = new string(charray);

string str2 = "Another string.";

Console.WriteLine(str1);

Console.WriteLine(str2);

}

}

**Operating on Strings**

* The **string** class contains several methods that operate on strings.
* The **string** type also includes the **Length** property, which contains the length of the string.
* To obtain the value of an individual character of a string, you simply use an index. For example:
  + string str = "test";
  + Console.WriteLine(str[0]);
* This displays “t”, the first character of “test”. Like arrays, string indexes begin at zero.
* One important point, however, is that you cannot assign a new value to a character within a string using an index. An index can only be used to obtain a character.

|  |  |
| --- | --- |
| **Method** | **Description** |
| static string Copy(string *str*) | Returns a copy of *str.* |
| int CompareTo(string *str*) | Returns less than zero if the invoking string is less than *str,* greater than zero if the invoking string is greater than *str,* and zero if the strings are equal |
| int IndexOf(string *str*) | Searches the invoking string for the substring specified by *str.* Returns the index of the first match, or –1 on failure. |
| int LastIndexOf(string *str*) | Searches the invoking string for the substring specified by *str.* Returns the index of the last match, or –1 on failure. |
| string ToLower( ) | Returns a lowercase version of the invoking string. |
| string ToUpper( ) | Returns an uppercase version of the invoking string. |

**Arrays of Strings:**

Like any other data type, strings can be assembled into arrays. For example:

// Demonstrate string arrays.

**using System;**

**class** StringArrays

{

**public static void Main()**

{

string[] str = { "This", "is", "a", "test." };

Console.WriteLine("Original array: ");

for(int i=0; i < str.Length; i++)

Console.Write(str[i] + " ");

Console.WriteLine("\n");

// Change a string.

str[1] = "was";

str[3] = "test, too!";

Console.WriteLine("Modified array: ");

for(int i=0; i < str.Length; i++)

Console.Write(str[i] + " ");

}

}