عنوان مضمون

Visual Programming-I

توسط : صفری

خزان1397

Strings

- Your use of strings so far has consisted of writing strings to the console, reading strings from the console, and concatenating strings using the + operator.

- To start with, **a string type variable can be treated as a read-only array of char variables.** This means that you can access individual characters using syntax like the following:

```
string myString = "A string";
char myChar = myString[1];
```

- To get a char array that you can write to you can use the following code. This uses the ToCharArray() command of the array variable:

```
string myString = "A string";
char[] myChars = myString.ToCharArray();
```

- Then you can manipulate the char array the standard way. You can also use strings in foreach loops, as shown here:

```
foreach (char character in myString)
{       WriteLine($"{character}");
}
```

- As with arrays, you can also get the number of elements using myString.Length. This gives you the number of characters in the string:

```
string myString = ReadLine();
WriteLine($"You typed {myString.Length} characters.");
```

- Other basic string manipulation techniques use commands with a format similar to this <*string*> .ToCharArray() command.

- Two simple, but useful, ones are <*string*>.**ToLower()** and <*string*> .**ToUpper().**
- These enable strings to be converted into lowercase and uppercase, respectively.
‘
- To see why this is useful, consider the situation in which you want to check for a specific response from a user—for example, the string yes. If you convert the string entered by the user into lowercase, then you can also check for the strings YES, Yes, yeS, and so on—you saw an example of this in the previous chapter:

```
string userResponse = ReadLine();
if (userResponse.ToLower() == "yes")
{       // Act on response.
}
```

- This is an important point to remember, because just writing

```
userResponse.ToLower();
```

doesn’t actually achieve very much!

- This command, like the others in this section, doesn’t actually change the string to which it is applied. Instead, combining this command with a string results in the creation of a new string,

```
userResponse = userResponse.ToLower();
```

- What if the user accidentally **put an extra space at the beginning or end of the input**? In this case, the preceding code won't work.
- You need to trim the string entered, which you can do using the <*string*>.Trim() command:

```
string userResponse = ReadLine();
userResponse = userResponse.Trim();
if (userResponse.ToLower() == "yes")
{          // Act on response.
}
```

- The preceding code is also able to detect strings like this:

```
"    YES"
"Yes "
```

- You can also use these commands to remove any other characters, by specifying them in a char array, for example:

```
char[] trimChars = {' ', 'e', 's'};
string userResponse = ReadLine();
userResponse = userResponse.ToLower();
userResponse = userResponse.Trim(trimChars);
if (userResponse == "y")
{           // Act on response.
}
```

- this will result in the detection of strings such as

  "Yeeeees"

  " y"

- and so on.

- You can also use the **<*string*>.TrimStart()** and **<*string*>.TrimEnd()** commands, which will **trim spaces from the beginning and end** of a string, respectively. These can also have char arrays specified.
- You can use two other string commands to manipulate the spacing of strings:
- You use them as follows:**<*string*>.PadLeft() and <*string*>.PadRight(). They enable you to add spaces to the left or right of a string to force it to the desired length.**

<*string*>.PadX(<*desiredLength*>);

- Here is an example:

```
myString = "Aligned";
myString = myString.PadLeft(10);
```

- This would result in three spaces being added to the left of the word Aligned in myString. These

methods can be helpful when aligning strings in columns, which is particularly useful for positioning strings containing numbers.

- As with the trimming commands, you can also use these commands in a second way, by supplying the character to pad the string with. This involves a single char, not an array of chars as with trimming:
- myString = "Aligned";
- myString = myString.PadLeft(10, '-');

- This would add three dashes to the start of myString.

Try It Out

```
static void Main(string[] args)
{       string myString = "This is a test.";
        char[] separator = {' '};
        string[] myWords;
        myWords = myString.Split(separator);
        foreach (string word in myWords)
        {       WriteLine($"{word}");
        }
        ReadKey();
}
```

Home work

- Write a console application that accepts a string from the user and outputs each word of string with the characters in reverse order.