

# FUNDAMENTALS OF PROGRAMMING WITH C#

## STRING HANDLING

Liu Fan

[lsslf@nus.edu.sg](mailto:lsslf@nus.edu.sg)

Total:24

# Objectives

- Write a program that manipulates string for various purpose
- Write a program that is able to manipulate string by its characters

# Agenda

- String Operation
- String Properties
- String Methods
- String and Characters

- Strings are series of characters.
- Strings Constant never change and are indicated by double quotes.
  - Examples: “Institute of Systems Science” “monkey”
- Variable strings are a special type of variable that are capable of storing strings.
  - Declaration:  
string Country;

- Strings like numerals are capable of being manipulated or operated upon. The following are the types of operations that you can perform on Strings:
  - Operators for putting strings together.
  - Functions for pulling strings apart.
  - Functions for creating strings.
  - Functions for modifying strings.
  - Operators for finding out things about strings.

Note: In any of the above operators and functions you can use either a variable or constant string.

- Strings like numerals are capable of being manipulated or operated upon.
- There are many ways by which you can perform an operation on a string in C#
  - Use of Operators
  - Use of methods by treating "string" as an object
  - Use of static methods in the String class
- We would confine to the first two approaches in this lesson.

- Use of operator on a string.
  - Comparison
    - Two strings can be compared with the help of relational operators just the way we do it for the numeric.
    - Example
      - Is ("Monkey" == "Donkey") ?      Answer is NO.
      - Is ("Monkey" != "Donkey") ?      Answer is YES
      - Is ("Monkey" == "monkey") ?      Answer is NO
    - The above Boolean (relational) expressions is usually used in "if" statements or while/for/select statements.
    - You cannot use the other relational operators like > and < on the strings.
    - Example
      - Is ("Monkey" > "Donkey") ?      **This will generate an error!**

# String Operations

- Use of operator on a string.
  - Concatenation
    - Strings are concatenated (joined together) using a + operator.
    - Example:
      - “Fundamental of ” + “Programming”   =>   “Fundamental of Programming”
      - “Hickory “ + “Dickory ” + “Dock”   => “Hickory Dickory Dock”



# String Properties

- Length
  - The Length property of the string provides the length of the string object
  - Length is read only property
  - Example:

Program:

```
string s = "ISS";  
Console.WriteLine(s.Length);
```

Output:

3

# String Methods

- Trim

- The Trim method if used without an argument, removes white spaces at the beginning and end of the string.

Example:

Program:

```
string s = "  abc  ";  
Console.WriteLine("*" + s + "*");  
string r = s.Trim()  
Console.WriteLine ("*" + r + "*");
```

Output:

```
*  abc  *  
*abc*
```

# String Methods

- Trim(chars)
  - The Trim method if used with an argument array of characters, removes specified Unicode characters at the beginning and end of the string.

Example:

Program:

```
string s = "$%$$abc%s%$";  
char[] c = new char[] { '$', '%' };  
Console.WriteLine ("*" + s + "*");  
string r = s.Trim(c);  
Console.WriteLine ("*" + r + "*");
```

Output:

```
*$%$$abc%s%$*  
*abc%s*
```

- TrimEnd(chars)
- TrimStart(chars)
- The above two methods work the same way as the Trim method with a character array argument. The only difference is:
  - TrimEnd trims the specified characters at the end of the string only leaving the head of the string untouched
  - TrimStart trims the specified characters at the beginning of the string only leaving the tail untouched

# String Methods

- Substring(int,int)
  - Substring method of the string extracts a part of the string
  - The Substring method uses two integer arguments. The first argument indicates the starting position and second argument the length

Example:

Program:

```
string s = "fundamental";  
string r = s.Substring(4,3);  
Console.WriteLine(r);  
Console.WriteLine (s.Substring(6,4));
```

Output:

```
ame  
enta
```

# String Methods

- Equals(string)
  - Equals method compares the string with the argument and returns a true if the two strings are same else it returns a false.

Example:

Program:

```
string s = "abc";  
string r = "xyz";  
if ( s.Equals(r) )  
    Console.WriteLine( "s and r are same");  
else  
    Console.WriteLine( "s and r are not same");  
if (s.Equals("abc"))  
    Console.WriteLine(" s is \"abc\" ");
```

Output:

```
s and r are not same  
s is "abc"
```

# String Methods

- `s1.CompareTo(s2)`
  - Compares this instance with a specified String object and indicates whether this instance precedes, follows, or appears in the same position in the sort order as the specified string.
  - Less than zero: s1 precedes s2
  - Zero: s1 has the same position in the sort order as s2
  - Greater than zero: s1 follows s2

Example:

Program:

```
string s1 = "abc";  
string s2 = "xyz";  
Console.WriteLine("s1.CompareTo(s2):" + s1.CompareTo(s2));  
string s3 = "football";  
string s4 = "foofoo";  
Console.WriteLine("s3.CompareTo(s4):" + s3.CompareTo(s4));  
string s5 = "foot";  
string s6 = "foot";  
Console.WriteLine("s5.CompareTo(s6):" + s5.CompareTo(s6));
```

Output:

```
s1.CompareTo(s2):-1  
s3.CompareTo(s4):1  
s5.CompareTo(s6):0
```

# String Methods

- `Insert(int, string)`
  - Insert method inserts into the string another string (given as second argument) from position given in the first argument and returns the new value.

Example:

Program:

```
string s = "Institute Systems Science";  
string r = s.Insert(10, "of ");  
Console.WriteLine(s);  
Console.WriteLine(r);
```

Output:

```
Institute Systems Science  
Institute of System Science
```



# String Methods

- `PadLeft(int n, char c)`
- `PadRight(int n, char c)`

Example:

Program:

```
string s = "ABC";  
string r = s.PadLeft(7, 'c');  
Console.WriteLine(s);  
Console.WriteLine(r);  
Console.WriteLine(s.PadRight(6,'z'));
```

Output:

```
ABC  
ccccABC  
ABCzzz
```

# String Methods

- ToUpper()
- ToLower()
  - ToUpper method returns the string value in Upper Case
  - ToLower method returns the string value in Lower Case

Example:

Program:

```
string s = "Institute Systems Science ";  
string r = s.ToUpper();  
Console.WriteLine(s);  
Console.WriteLine(r);  
Console.WriteLine(s.ToLower());
```

Output:

```
Institute Systems Science  
INSTITUTE SYSTEMS SCIENCE  
institute systems science
```

# String Methods

- Split(char c)
  - Split a string into an array of substrings separated by a character such as a comma.
- Split(char[])
  - Splits a string based on multiple characters

Example:

Program:

```
string authors = "Henry He, Chris Chew, Praveen Kumar ";  
string[] authorsList = authors.Split(',');  
for (int i = 0; i <= authorsList.Length - 1; i++)  
    Console.WriteLine(authorsList[i]);
```

Output:

```
Henry He  
Chris Chew  
Praveen Kumar
```

# String Methods

- Contains(String)
  - Returns a value indicating whether a specified substring occurs within this string.

Example:

Program:

```
string s = "Institute Systems Science";  
string s1 = "Systems";  
Console.WriteLine(s.Contains(s1));
```

Output:

True

# String and Characters

- Characters that makes up a string can be read individually using the index
  - The index starts with 0. The first character of the string has the index of 0
  - But they cannot be modified

```
string s="Hello";  
Console.WriteLine(s[1]);  
  
s[1] = 'b';
```

e

Error!

- If you want to be able to modify each of the character, you can check StringBuffer class

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/strings/>

# String and Characters

- string is a data type that store string/sequence of zero to many characters. E.g.: "", "a", "abc"
- char is a data type that store ONE character.
  - E.g.: 'a' or 'b'
  - These are not valid: " and 'ab'
- char is written with a single quote '
- If `s` is a string, `s[3]` returns similar value as `s.Substring(3,1)` but different data types.

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/char>

# Interesting facts about char

- char can be implicitly converted into int
  - This is inherited from the fact that in C language, char is also treated as number
  - But the reverse is not true.

```
char x = 'a';  
Console.WriteLine(x);  
int y = x;  
Console.WriteLine(y);  
Console.WriteLine((char)y);  
Console.WriteLine('9' - '0');  
Console.WriteLine('z' - 'a');
```

```
a  
  
97  
a  
9  
25
```

# Summary

- string is an object that has useful property like Length and methods that can be used to manipulate strings
- string.Split() is one of the most useful function for parsing a string
  - Breaking a string into words
  - Read a comma separated values (CSV)
- Individual characters in a string can be individually addressed
- Note some differences between string and char.
  - String is more flexible and easier to use