## Lab 9 : Strings

1. Write a method that prints out every character in a string that is passed to it as an argument.
2. Write a method that prints out every third character in a string that is passed to it as an argument.
3. Write a method that receives a string as an argument and returns the number of times the character ‘x’ or ‘y’ appears in the string.
4. Write a method that that accepts a string variable as an argument and returns a new string made up of every alternate 3 characters (e.g “**abc**def**ghi**j” will return “abcghi”);
5. Write a method that returns true if the string that is passed to it as an argument starts with a numeric character.
6. Write a method that accepts a string variable as an argument and returns a new string with a ‘-‘ inserted after every third character (e.g. “ 97755512312” will return “977-555-123-12”)
7. Write a method that accepts a string variable as an argument and returns a new string with the characters ‘$’, ‘%’, ‘,’ ‘ ‘ stripped out of it. (e.g. “abc$$kh% gg,” will return “abckhgg”
8. Write a method that accepts a string variable as an argument and returns *true* if the argument ends with “.com”. Otherwise the method should return *false*.
9. Develop a program to send and receive a coded message
10. Write a method:

static string Encrypt(string str)

That returns the string str with each adjacent pair of characters swapped. For example:

Encrypt(“my secrets”) returns “yms ceerst”

(Hint: an string can be thought of as an array of characters

Hint: use a for loop to step through the characters in the string)

**Chapter 3 - String Handling**

The String class has numerous methods that help you in working with the string objects. There are lots of times when writing a program we will want to manipulate strings, for instance**.**

* Comparing two strings
* Convert a string from uppercase to lower case
* Check the length of a string, say a password
* Check that a particular part of a string contains a particular value – for example is the first character in a password alphabetic
* Become a spy and do some encryption/decryption of secret messages (exciting!)
* Can you think of some examples?

**The following table provides some of the most commonly used methods −**

|  |  |
| --- | --- |
| **No.** | **Methods & Description** |
| 1 | **public static int Compare(string strA, string strB)**  Compares two specified string objects and returns an integer that indicates their relative position in the sort order. |
| 2 | **public static int Compare(string strA, string strB, bool ignoreCase )**  Compares two specified string objects and returns an integer that indicates their relative position in the sort order. However, it ignores case if the Boolean parameter is true. |
| 3 | **public static string Concat(string str0, string str1) // overloaded with 3 others**  Concatenates two string objects. The other deal with 3 and 4 bjects |
| 6 | **public bool Contains(string value)**  Returns a value indicating whether the specified String object occurs within this string. |
| 7 | **public static string Copy(string str)**  Creates a new String object with the same value as the specified string. |
| 8 | **public void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int count)**  Copies a specified number of characters from a specified position of the String object to a specified position in an array of Unicode characters. |
| 9 | **public bool EndsWith(string value)**  Determines whether the end of the string object matches the specified string. |
| 10 | **public bool Equals(string value)**  Determines whether the current String object and the specified String object have the same value. |
| 11 | **public static bool Equals(string a, string b)**  Determines whether two specified String objects have the same value. |
| 12 | **public static string Format(string format, Object arg0)**  Replaces one or more format items in a specified string with the string representation of a specified object. |
| 13 | **public int IndexOf(char value)**  Returns the zero-based index of the first occurrence of the specified Unicode character in the current string. |
| 14 | **public int IndexOf(string value)**  Returns the zero-based index of the first occurrence of the specified string in this instance. |
| 15 | **public int IndexOf(char value, int startIndex)**  Returns the zero-based index of the first occurrence of the specified Unicode character in this string, starting search at the specified character position. |
| 16 | **public int IndexOf(string value, int startIndex)**  Returns the zero-based index of the first occurrence of the specified string in this instance, starting search at the specified character position. |
| 17 | **public int IndexOfAny(char[] anyOf)**  Returns the zero-based index of the first occurrence in this instance of any character in a specified array of Unicode characters. |
| 18 | **public int IndexOfAny(char[] anyOf, int startIndex)**  Returns the zero-based index of the first occurrence in this instance of any character in a specified array of Unicode characters, starting search at the specified character position. |
| 19 | **public string Insert(int startIndex, string value)**  Returns a new string in which a specified string is inserted at a specified index position in the current string object. |
| 20 | **public static string PadRight(int value) // also one for PadLeft**  Return a string that’s right aligned and padded on the left with spaces so it is the specified length |
| 21 | **public static string Join(string separator, params string[] value)**  Concatenates all the elements of a string array, using the specified separator between each element. |
| 22 | **public static string Join(string separator, string[] value, int startIndex, int count)**  Concatenates the specified elements of a string array, using the specified separator between each element. |
| 23 | **public int LastIndexOf(char value [,start index])**  Returns the zero-based index position of the last occurrence of the specified Unicode character within the current string object. |
| 24 | **public int LastIndexOf(string value [,start index])**  Returns the zero-based index position of the last occurrence of a specified string within the current string object. |
| 25 | **public string Remove(int startIndex)**  Removes all the characters in the current instance, beginning at a specified position and continuing through the last position, and returns the string. |
| 26 | **public string Remove(int startIndex, int count)**  Removes the specified number of characters in the current string beginning at a specified position and returns the string. |
| 27 | **public string Replace(char oldChar, char newChar)**  Replaces all occurrences of a specified Unicode character in the current string object with the specified Unicode character and returns the new string. |
| 28 | **public string Replace(string oldValue, string newValue)**  Replaces all occurrences of a specified string in the current string object with the specified string and returns the new string. |
| 29 | **public string[] Split(params char[] separator)**  Returns a string array that contains the substrings in the current string object, delimited by elements of a specified Unicode character array. |
| 30 | **public string[] Split(char[] separator, int count)**  Returns a string array that contains the substrings in the current string object, delimited by elements of a specified Unicode character array. The int parameter specifies the maximum number of substrings to return. |
|  | Substring(StartIndex[,length])  Return a string that starts at the specified position and has the specified length |
| 31 | **public bool StartsWith(string value)**  Determines whether the beginning of this string instance matches the specified string. |
| 32 | **public char[] ToCharArray()**  Returns a Unicode character array with all the characters in the current string object. |
| 33 | **public char[] ToCharArray(int startIndex, int length)**  Returns a Unicode character array with all the characters in the current string object, starting from the specified index and up to the specified length. |
| 34 | **public string ToLower()**  Returns a copy of this string converted to lowercase. |
| 35 | **public string ToUpper()**  Returns a copy of this string converted to uppercase. |
| 36 | **public string Trim() :** remove all leading and trailing white space characters from current strin |

// Demo of some string functions

class Program

{

static void Main(string[] args)

{

string address = "pat ryan,sligo";

// write out the first character

Console.WriteLine("Test String = {0}", address);

Console.WriteLine("The first character is : {0}",address[0]); // 0 is the index of first character

// write out the last character - note the use of Length – 1

// here we make use of the Length property to get the number of chars

Console.WriteLine("The last character is {0}", address[address.Length - 1]);

// write out the individual characters that make up a string

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

{

Console.WriteLine("Character {0} = {1}",i, address[i]);

}

// check if string starts with "pat"

bool checkStart = address.StartsWith("pat");

Console.WriteLine("Does the string start with \"pat\" ? - {0}",checkStart);

// append the string "Ireland" to the end of the string

address = address.Insert(address.Length, ",Ireland");

Console.WriteLine("New string with \"Ireland\" appended to the end - {0}", address);

// write a full name as initials

string fullName = "John Pat Ryan";

string firstInitial = fullName.Substring(0, 1);

int spacePos = fullName.IndexOf(" ");

string secondInitial = fullName.Substring(spacePos + 1, 1);

int secondSpace = fullName.IndexOf(" ", spacePos + 1);

string thirdInitial = fullName.Substring(secondSpace + 1, 1);

string newName = firstInitial + "." + secondInitial + "." + thirdInitial;

Console.WriteLine("Initials of \"John Pat Ryan\" = {0}",newName);

// remove ' ', '$', '," and '\*' from a string

// this has a 2 bugs... fix it

string funnyMessage = "Brexit is $ a great idea,\*";

int l = funnyMessage.Length - 1;

for (int i = 0; i < funnyMessage.Length - 1; i++)

{

Console.WriteLine(funnyMessage[i]);

if (funnyMessage[i] == ' ' || funnyMessage[i] == '$' || funnyMessage[i] == ',' || funnyMessage[i] == '\*')

{

funnyMessage = funnyMessage.Remove(i, 1);

}

}

Console.WriteLine("\"Brexit is $ a great idea, \*\" with non alphabetic chars removed = {0}",funnyMessage); }

}