Лабораторная работа № 3

Тема: Работа со строками

Цель: Сохранение и манипулирования текстовыми данными. Использование массивов и коллекций.

теоретические сведения

Строка в C # представляет собой один или несколько символов, объединенных в группу и объявленных с помощью ключевого слова string, которое является ускоренным методом языка C # для класса System.String. В отличие от массивов символов в C или C ++, строки в C # гораздо проще в использовании и менее склонны к ошибкам программирования.

Строка в C # - это объект типа String значением которого является текст. Каждый символ в строке занимает 2 байта (в .NET по умолчанию используется Unicode).

**Объявления и инициализация строк**

Создать строку можно несколькими способами:

string s; // объявление без инициализации

string s = null; // объявление и инициализация значением null

string s = "Hello, World!"; // объявление и инициализация текстовым значением

string s = new string ( '', 20); // Конструктор создает строку из 20 пробелов

char [] a = { '0', '0', '0'}; // Массив для инициализации строки

string s = new string (а) // Создание из массива символов

**Работа со строками. Методы класса String**

Для строк определены следующие операции:

назначения =;

проверка на равенство ==;

проверка на неравенство! =;

обращение по индексу [];

конкатенация строк +.

== Строки равны, если имеют одинаковое количество символов и совпадают посимвольно.

[] Обращаться к отдельному элементу строки по индексу можно только для получения значения, но не для его изменения. Это связано с тем, что строки типа string относятся к так называемым неизменных типов данных. Методы, изменяющие содержимое строки, на самом деле создают новую копию строки. Ненужные "старые" копии автоматически удаляются при сборке мусора.

**Объединение (конкатенация) строк**

Строки можно объединять между собой, как показано в следующем примере:

string s1 = "Строка более";

string s2 = "суммы его символов.";

// Объединяя S1 и S2, мы на самом деле создаем новый

// Объект и сохраняем его в s1, заменяя ссылки

// На исходный объект ссылкой на новый объект

s1 + = s2;

System.Console.WriteLine (s1)

// Результат: Строка больше, чем сумма его символов.

Срочные объекты являются неизменными: после создания их нельзя изменить. Методы, работающие со строками, возвращают новые строчные объекты. Поэтому с целью повышения производительности большие объемы работы по объединению строк или иные операции следует выполнять в классе StringBuilder, как показано в дальше в примерах кода.

Таблица 3.1. Некоторые свойства и методы класса System.String

|  |  |
| --- | --- |
| **Назва** | **Призначення** |
| Length | Властивість. Дозволяє отримати кількість символів в рядку. |
| Concat() | Дозволяє з'єднати декілька рядків або змінних типу object. |
| CompareTo() | Дозволяє порівняти два рядки. В разі рівності рядків результат виконання функції дорівнює нулю. При позитивному значенні функції більшим є рядок, для якого викликався метод. |
| Copy() | Створює нову копію існуючого рядка. |
| Format() | Застосовується для форматування рядка з використанням різних примітивів (рядків і числових даних) і підстановлювальних виразів вигляду {0}. |
| Insert() | Дозволяє вставити один рядок всередину існуючого. |
| Remove() | Видаляє символи в рядку. |
| Replace() | Замінює символи в рядку. |
| ToUpper() | Переводять всі символи рядка в верхній реєстр. |
| ToLower() | Переводять всі символи рядка в нижній реєстр. |
| Chars | Дозволяє отримати символ, що знаходиться в певній позиції рядка. |
| Join() | Створює рядок, сполучаючи задані рядки і розділяючи їх рядком-роздільником. |
| Replace() | Замінює один символ рядка іншим. |
| Split() | Повертає масив рядків з елементами - підрядками основного рядка, між якими знаходяться символи-роздільники. |
| Substring() | Дозволяє отримати підрядок основного рядка, заданої довжини, що починається з певного символу. |
| Trim() | Видаляє пропуски або набір заданих символів на початку і кінці основного рядка. |
| ToCharArray() | Створює масив символів і вставляє в нього символи початкового рядка. |

**Метод ToString ()**

Все встроенные типы данных C # предоставляют преобразования метод ToString, преобразующий значение указанной переменной преобразования в строку. Этот метод может быть использован для преобразования числовых значений в строки следующим образом:

int year = 1999;

string msg = "Єва народилася в " + year.ToString();

System.Console.WriteLine(msg); // Результат: "Єва народилася в 1999"

**изменение регистра**

Для переключения регистра букв в строке (сделать их большими или малыми) следует использовать ToUpper () или ToLower (), как показано в следующем примере.

// string s6 = "Битва при Гастінгсі, 1066";

System.Console.WriteLine(s6.ToUpper());

// outputs "БИТВА ПРИ ГАСТІНГСІ 1066"

System.Console.WriteLine(s6.ToLower());

// outputs "битва при гастінгсі 1066"

**Доступ к отдельным знаков**

К отдельным знаков, содержащихся в строке, можно получить доступ с помощью таких методов как, например, Substring, Replace и IndexOf.

**Получение подстроки методом Substring:**

string s3 = "Visual C# Express";

System.Console.WriteLine(s3.Substring(7, 2));

// Результат: "C#"

**Замена подстроки методом Replace:**

System.Console.WriteLine(s3.Replace("C#", "Basic"));

// Результат: "Visual Basic Express"

**Получение номера первого вхождения символа в строку:**

// Значення індексів починається з нуля

int index = s3.IndexOf("C");

// index = 7

Доступ к отдельным знаков в строке возможен с помощью индекса, как показано в следующем примере.

string s5 = "Друк у зворотному напрямку";

for (int i = 0; i < s5.Length; i++)

{

System.Console.Write(s5[s5.Length - i - 1]);

}

// Результат: "укмярпан умонторовз у курД"

**Escape-символы**

Строки могут содержать escape-знаки, такие как "\ n" (новая строка) и "\ t" (табуляция). строка:

string columns = "Стовпчик 1\tСтовпчик 2\tСтовпчик 3";

//Результат: Стовпчик 1 Стовпчик 2 Стовпчик 3

string rows = "Рядок 1\r\nРядок 2\r\nРядок 3";

/\* Результат:

Рядок 1

Рядок 2

Рядок 3

\*/

string title = "\" The \ u00C6olean Harp \ ", by Samuel Taylor Coleridge";

// Результат: "The Æolean Harp", by Samuel Taylor Coleridge

Если нужно добавить в строку обратную косую черту, перед ней нужно поставить еще одну обратную косую черту. Следующую строку:

string fff = "\\\\ My Documents \\";

эквивалентный строке:

\\ My Documents \

символ @

Символ @ указывает, что при создании строк следует игнорировать escape-знаки и переносы строк. Наступнидварядкы идентичны.

string p1 = "\\\\ My Documents \\ MyFiles \\";

string p2 = @ "\\ My Documents \ MyFiles \";

Еще примеры применения символа @:

string filePath = @ "C: \ Users \ scoleridge \ Documents \";

// Результат: C: \ Users \ scoleridge \ Documents \

string quote = @ "Her name was" "Sara". "";

// Результат: Her name was "Sara".

Таблица 3.2. Escape-символов языка С #

| **Escape-Послідовність** | **Ім'я символу** | **Кодування Юнікоду** |
| --- | --- | --- |
| \’ | Одинарні лапки | 0×0027 |
| \" | Подвійні лапки | 0×0022 |
| \\ | Зворотна коса риса | 0x005C |
| \0 | Null | 0×0000 |
| \a | ALERT | 0×0007 |
| \b | Backspace | 0×0008 |
| \f | FORM FEED | 0x000C |
| \n | Новий рядок | 0x000A |
| \r | Повернення каретки | 0x000D |
| \t | Горизонтальна табуляція | 0×0009 |
| \U | Escape-Послідовність Юнікоду для пар символів-заступників. | \Unnnnnnnn |
| \u | Escape-Послідовність Юнікоду | \u0041 = "A" |
| \v | Вертикальна табуляція | 0x000B |
| \x | Escape-Послідовність Юнікоду аналогічна "\u", за винятком рядків зі змінною довжиною. | \x0041 = "A" |

**Класс StringBuilder и его методы**

Иногда следует избегать ситуаций, когда в результате выполнения операции создается новая строка, поскольку это связано с дополнительными затратами памяти и других ресурсов компьютера при выполнении операции.

C # содержит специальный класс StringBuilder, используя который можно избежать создания копий строк при их обработке. Все изменения, вносимые в объект данного класса, немедленно отражаются в нем, что эффективнее, чем работа с копиями строки.

Основной операцией, которая чаще всего используется классом StringBuilder, является операция сложения в строку содержания. Для этого существует метод Append. Следующий код добавляет одну строку к другому и выводит результат на консоль. При этом изменяется оригинал строки, копия не создается:

StringBuilder sb = new StringBuilder ( "На следующей неделе у нас модульной контроль»);

sb.Append ( "нужно защитить лабатораторни работы")

Сonsole.Write (sb)

Кроме добавления класс StringBuilder содержит другие методы, наиболее значимые из которых приведены ниже. После того, как все необходимые действия, связанные с обработкой строки, были выполнены, необходимо вызвать метод ToString () для преобразования содержания объекта в обычный тип данных string.

Таблица 3.3. Некоторые методы класса StringBuilder

|  |  |
| --- | --- |
| **Назва** | **Призначення** |
| Append | Додавання заданого рядка в кінець рядка об'єкту. |
| AppendFormat | Додавання заданого форматованого рядка (рядка, що містить управляючі символи) в кінець рядка об'єкту. |
| CopyTo | Копіювання символів заданого сегменту рядка в задані комірки масиву символів. |
| Insert | Додавання рядка в задану позицію рядка об'єкту. |
| Remove | Видалення заданої кількості символів з рядка об'єкту |
| Replace | Заміна заданого символу або рядка об'єкту на інший заданий символ або рядок. |

При интенсивной работе со строками рекомендуется использовать класс StringBuilder, поскольку это позволяет уменьшить накладные расходы, связанные с созданием копии строки при выполнении каждой операции.

Пример работы с классом StringBuilder:

string question = "hOW DOES mICROSOFTwORD DEAL WITH THE cAPSlOCK KEY?";

//запис змінної типу String в змінну типу StringBuilder

System.Text.StringBuilder sb = new System.Text.StringBuilder(question);

for (int j = 0; j <sb.Length; j++)

{

if (System.Char.IsLower(sb[j]) == true)

sb[j] = System.Char.ToUpper(sb[j]);

elseif (System.Char.IsUpper(sb[j]) == true)

sb[j] = System.Char.ToLower(sb[j]);

}

// Збереження відредагованого рядка

string corrected = sb.ToString();

System.Console.WriteLine(corrected);

// Результат: Howdoes Microsoft Word deal with the CapsLock key?

пример 3.1

Рассмотрим работу со строками в C # на примере простого шифрования текста - так называемой литорея.

Литорея (от littera) - тайнопись, род шифрованного письма, использовался в древнерусской рукописной литературе. Известная литорея двух типов: простая и мудрая. Простая, иначе называется тарабарском грамотой, заключается в следующем: поставив согласные буквы в два ряда, в следующем порядке:

б

в

г

д

же

с

к

л

м

н

ч

ш

ч

ц

х

ф

т

с

г.

п

записывают в зашифрованном письме верхние буквы вместо нижних и наоборот - нижние вместо верхних, причем гласные остаются без изменения; так, например, лсошамь = словарь и т.п.

Мудрая литорея предполагает более сложные правила подстановки. В различных вариантах, которые дошли до нас, используются подстановки целых групп букв, а также числовые комбинации: каждой согласной букве ставится в соответствие число, а затем происходят арифметические действия над полученной последовательностью чисел (например, ко всем числам добавлялась некоторая константа - ключ).

По своей сути литорея является шифром простой замены легко дешифруется современными методами.

Создадим Windows-приложение с графическим интерфейсом пользователя, реализует алгоритм шифрования простой литорея.

Для реализации на С # работы с текстом используем элемент управления richTextBox - поле форматированного текста.

Пример исходного кода, реализующего простую литорею:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

// Программа для кодировки текста

// методом простой литорея

public partial class Form1: Form

{

const string lit1 = "АБВГДЕЖЗИЙКЛМНОП";

const string lit2 = "РСТУФХЦЧШЩЪЫЬЭЮЯ";

public Form1 ()

{

InitializeComponent ();

}

// Обработка нажатия на кнопку "шифровать"

private void btnGo\_Click (object sender, EventArgs e)

{

// исходный текст:

string text = rtxSource.Text;

// литорейний текст:

string litorea = "";

// длина текста:

int len = text.Length;

// шифруем - заменяем буквы парными:

for (int i = 0; i <len; ++ i) {

// очередная буква:

char chr = text [i];

chr = char.ToUpper (chr)

int n = lit1.IndexOf (chr)

if (n> -1) // верхняя буква

// заменяем нижней:

litorea + = lit2 [n];

else

{

n = lit2.IndexOf (chr)

if (n> -1) // нижняя буква

// заменяем верхней:

litorea + = lit1 [n];

else // буква Е или другой символ:

litorea = litorea + chr;

}

}

// выводим зашифрованный текст во втором текстовом поле:

rtxTarget.AppendText (litorea)

}

// очистить текстовые окна

private void btnClear\_Click (object sender, EventArgs e)

{

rtxSource.Clear ();

rtxTarget.Clear ();

}

}

Пример скриншота программы приведен на рисунке 1.

\*\*\*\*\*\*\* 1

Рисунок 3.1 - Окно программы «литорея», осуществляющая шифрование текста

Задания к лабораторной работе:

вариант 1

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает все слова в тексте в обратном порядке.

вариант 2

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает с 1-го только те слова в которых буква «р».

вариант 3

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает с 1-го только те слова, которые начинаются на гласную букву.

вариант 4

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст украинскими буквами, в 2-е программа записывает транслитерацию текста в латинский алфавит.

вариант 5

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает с 1-го только те предложения, которые не содержат ком и двоеточий.

вариант 5

Написать программу, в которой есть текстовое поле richTextBox и textBox. В поле richTextBox пользователь вводит текст, а в поле textBox программа записывает число, соответствующее количеству слов в тексте с 5-ти букв.

вариант 6

Написать программу, в которой есть два поля richTextBox и одно- textBox. В 1-е поле richTextBox пользователь вводит текст, в поле textBox вводит состав. В 2-е поле richTextBox выводятся все слова с этим составом.

вариант 7

Написать программу, в которой есть поле richTextBox и 2 поля textBox. В richTextBox пользователь вводит текст, в поля textBox - слово из текста и слово для замены. Замена производится по нажатии кнопки «Заменить».

вариант 8

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает длинное слово в тексте и число его повторов в тексте.

вариант 9

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает сначала все восклицательные предложения, затем вопросительные, потом все остальные.

вариант 7

Написать программу, в которой есть поле richTextBox и 2 поля textBox. В richTextBox вводится текст, а в I поле textBox - слово для поиска. Во II поле textBox выводится сколько раз данное слово встречается в тексте.

вариант 8

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает все предложения в тексте в обратном порядке.

вариант 9

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает введенный текст меняя во всех словах первую и последнюю буквы местами.

вариант 10

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает предложения из введенного текста, в котором чаще всего встречается буква «О».

вариант 11

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает только те предложения, в которых есть цифры и количество таких предложений.

вариант 12

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает только те предложения, которые начинаются словом с одной буквы.

вариант 13

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает только те слова, которые начинаются и заканчиваются на одинаковую букву.

вариант 14

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает вместо букв - их номер в алфавите, а вместо цифр - их название.

вариант 15

Написать программу, в которой есть два текстовых поля (richTextBox). В 1 текстовое поле пользователь вводит текст, 2-е программа записывает только те слова, количество букв в которых - простое число.

Контрольные вопросы:

1. Как инициализировать строчную переменную?

string s = null; // объявление и инициализация значением null

string s = "Hello, World!"; // объявление и инициализация текстовым значением

string s = new string ( '', 20); // Конструктор создает строку из 20 пробелов

char [] a = { '0', '0', '0'}; // Массив для инициализации строки

string s = new string (а); // Создание из массива символов

1. Какие вы знаете методы для работы со строками?

ToString(), ToInsert()….;

1. С помощью каких операторов можно изменить регистр букв в строке?

ToUpper();, ToLower();

1. Что такое кодировка Unicode?
2. Какие средства работы с отдельными символами предоставляет C #?

Char(), index();

1. Какое основное ограничение имеет класс string?

По производительности(копирует экземпляр а не изменяет его)

1. Как определить количество символов в строке?

Имя\_переменой.Length;

1. Чем отличаются классы String и StringBuilder?

Объекты этого класса можно изменять.

1. Как объединить два объекта StringBuilder?

При помощи метода Append();

1. Почему иногда для работы со строками лучше использовать класс StringBuilder?

Что бы не производить лишних операций по преобразованию и копированию.



