**Лабораторна робота № 3**

Тема: Робота з рядками

Мета: Збереження та маніпулювання текстовими даними. Використання масивів та колекцій.

**Теоретичні відомості**

Рядок в C# представляє собою один або декілька символів, об’єднаних у группу та оголошених за допомогою ключового слова string, яке є прискореним методом мови C# для класу System.String. На відміну від масивів символів в C або C++, рядки в C# набагато простіші у використанні і менш схильні до помилок програмування.

**Рядок** в C# – це *об'єкт типу String* значенням якого є текст. Кожний символ у рядку займає 2 байти (в .NET за замовчуванням використовується Unicode).

**Оголошення та ініціалізація рядків**

Створити рядок можна декількома способами:

string s; //оголошення без ініціалізації

string s = null; //оголошення та ініціалізація значенням null

string s = "Hello, World!"; //оголошення та ініціалізація текстовим значенням

string s = new string(' ', 20); // конструктор створює рядок з 20 пробілів

char[] а = { '0', '0', '0' }; // масив для ініціалізації рядка

string s = new string(а); // створення з масиву символів

**Робота з рядками. Методи класу String**

Для рядків визначені наступні операції:

1. призначення =;
2. перевірка на рівність ==;
3. перевірка на нерівність !=;
4. звернення по індексу [];
5. конкатенація рядків +.

**==** Рядки рівні, якщо мають однакову кількість символів і збігаються посимвольно.

**[]** Звертатися до окремого елементу рядка за індексом можна лише для отримання значення, але не для його зміни. Це пов'язано з тим, що рядки типу string відносяться до так званих незмінних типів даних. Методи, що змінюють вміст рядка, насправді створюють нову копію рядка. Непотрібні "старі" копії автоматично видаляються при збірці сміття.

**Об’єднання (конкатенація) рядків**

Рядки можна об’єднувати між собою, як показано в наступному прикладі:

string s1 = "Рядок більше,";

string s2 = "ніж сума його символів.";

// Об'єднуючи S1 і S2, ми насправді створюємо новий

// об'єкт і зберігаємо його в s1, замінюючи посилання

// на початковий об’єкт посиланням на новий об’єкт

s1 += s2;

System.Console.WriteLine(s1);

// Результат: Рядок більше, ніж сума його символів.

Строкові об'єкти є незмінними: після створення їх не можна змінити. Методи, що працюють з рядками, повертають нові рядкові об'єкти. Тому з метою підвищення продуктивності великі обсяги роботи по об'єднанню рядків або інші операції слід виконувати в класі StringBuilder, як показано в далі в прикладах коду.

**Таблиця 3.1. Деякі властивості і методи класу 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()**

Всі вбудовані типи даних C# надають метод ToString, перетворюючий значення вказаної змінної в рядок. Цей метод може бути використаний для перетворення числових значень у рядки наступним чином:

int year = 1999;

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

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

**Зміна регістра**

Щоб змінити регістр літер у рядку (зробити їх великими або малими) слід використовувати ToUpper() або ToLower(), як показано в наступному прикладі.

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 до окремих знаків**

До окремих знаків, що містяться в рядку, можна отримати доступ за допомогою таких методів як, наприклад, *Substring*, *Replace* та *IndexOf*.

Отримання підрядка методом Substring:

string s3 = "Visual C# Express";

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

// Результат: "C#"

Заміна підрядка методом Replace:

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

// Результат: "Visual Basic Express"

Одержання номера першого входження символу в рядок:

// Значення індексів починається з нуля

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

// index = 7

Доступ до окремих знаків в рядку можливий за допомогою індексу, як показано в наступному прикладі.

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-символи**

Рядки можуть містити escape-знаки, такі як "\ n" (новий рядок) і "\ t" (табуляція). Рядок:

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

Якщо потрібно додати в рядок зворотну скісну риску, перед нею потрібно поставити ще одну зворотну скісну риску. Наступний рядок:

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

еквівалентний рядку:

\\My Documents\

**Символ @**

Символ @ вказує, що при створенні рядків слід ігнорувати escape-знаки і переноси рядків. Наступнідварядки є ідентичними.

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

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

Ще приклади застосування символу **@:**

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

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

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

//Результат: Her name was "Sara."

**Таблиця 3.2. Escape-символів мови 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" |

**Клас StringBuilder та його методи**

Іноді слід уникати ситуацій, коли в результаті виконання операції створюється новий рядок, оскільки це пов'язано з додатковими витратами пам'яті та інших ресурсів комп'ютера при виконанні операції.

C# містить спеціальний клас StringBuilder, використовуючи який можна уникнути створення копій рядків при їх обробці. Всі зміни, що вносяться до об'єкту даного класу, негайно відображаються в ньому, що ефективніше, ніж робота з копіями рядка.

Основною операцією, яка найчастіше використовується класом **StringBuilder**, є операція додавання до рядка вмісту. Для цього існує метод **Append**. Наступний код додає один рядок до іншого і виводить результат на консоль. При цьому змінюється оригінал рядка, копія не створюється:

StringBuilder sb = new StringBuilder("Наступного тижня у нас модульній контроль");

sb.Append(", потрібно захистити лабатораторні роботи");

Сonsole.Write(sb);

Окрім додавання клас StringBuilder містить інші методи, найбільш значимі з яких наведені нижче. Після того, як усі необхідні дії, пов'язані з обробкою рядка, були виконані, необхідно викликати метод ToString() для перетворення вмісту об'єкту в звичайний тип даних string.

**Таблиця 3.3. Деякі методи класу 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 інтенсивній роботі з рядками рекомендується використовувати клас **StringBuilder**, оскільки це дозволяє зменшити накладні витрати, пов'язані із створенням копії рядка при виконанні кожної операції.

Приклад роботи з класом 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?

**Приклад 3.1**

Розглянемо роботу з рядками в С# на прикладі простого шифрування тексту – так званої *літореї*.

**Літорея** (від littera) – тайнопис, рід шифрованого листа, що використовувався в давньоруській рукописній літературі. Відома літорея двох типів: *проста* та *мудра.* Проста, інакше називається тарабарською грамотою, полягає в наступному: поставивши приголосні букви у два ряди, у порядку:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **б** | **в** | **г** | **д** | **ж** | **з** | **к** | **л** | **м** | **н** |
| **щ** | **ш** | **ч** | **ц** | **х** | **ф** | **т** | **с** | **р** | **п** |

записують у шифрованому листі верхні букви замість нижніх і навпаки – нижні замість верхніх, причому голосні залишаються без зміни; так, наприклад, *лсошамь* = *словник* і т.п.

*Мудра літорея* припускає більш складні правила підстановки. У різних варіантах, що дійшли до нас, використовуються підстановки цілих груп букв, а також числові комбінації: кожній приголосній букві ставиться у відповідність число, а потім відбуваються арифметичні дії над отриманою послідовністю чисел (наприклад, до всіх чисел додавалася деяка константа - ключ).

По своїй суті Літорея є шифром простої заміни, що легко дешифрується сучасними методами.

Створимо Windows-додаток з графічним інтерфейсом користувача, що реалізує алгоритм шифрування простою літореєю.

Для реалізації на С# роботи з текстом використовуємо елемент керування **richTextBox** – поле форматованого тексту.

**Приклад сирцевого коду, що реалізує просту літорею:**

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;

//Програма для кодування тексту

//методом простої літореї

public partial class Form1 : Form

{

const string lit1 = "АБВГДЕЖЗИЙКЛМНОП";

const string lit2 = "РСТУФХЦЧШЩЪЫЬЭЮЯ";

public Form1()

{

InitializeComponent();

}

//Обробка натиснення на кнопку "шифрувати"

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

{

//вхідний текст:

string text = rtxSource.Text;

//літорейний текст:

string litorea = "";

//довжина тексту:

int len = text.Length;

//шифруємо - заміняємо букви парними:

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

//чергова буква:

char chr= text[i];

chr = char.ToUpper(chr);

int n = lit1.IndexOf(chr);

if (n > -1) // верхня буква

//заміняємо нижньою:

litorea += lit2[n];

else

{

n = lit2.IndexOf(chr);

if (n > -1) //нижня буква

//заміняємо верхньою:

litorea += lit1[n];

else //буква Е або інший символ:

litorea = litorea + chr;

}

}

//виводимо зашифрований текст у другому текстовому полі:

rtxTarget.AppendText(litorea);

}

//очистити текстові вікна

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

{

rtxSource.Clear();

rtxTarget.Clear();

}

}

Приклад скриншоту програми наведено на рисунку 1.

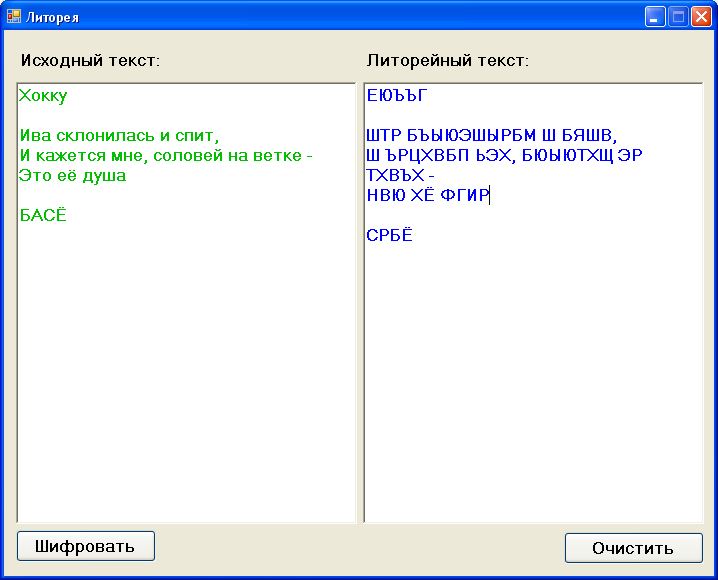


Рисунок 3.1 – Вікно програми «Літорея», що здійснює шифрування тексту

**Завдання до лабораторної роботи:**

**Варіант 1**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує всі слова в тексті у зворотному порядку.

**Варіант 2**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує з 1-го тільки ті слова у яких є літера «р».

**Варіант 3**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує з 1-го тільки ті слова, що починаються на голосну букву.

**Варіант 4**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст українськими літерами, в 2-ге програма записує транслітерацію тексту в латинський алфавіт.

**Варіант 5**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує з 1-го тільки ті речення, які не містять ком та двокрапок.

**Варіант 5**

Написати програму, у якій є текстове поле richTextBox та textBox. В поле richTextBox користувач вводить текст, а в поле textBox програма записує число, що відповідає кількості слів в тексті з 5-ти букв.

**Варіант 6**

Написати програму, у якій є два поля richTextBox та одне­­– textBox. В 1-ше поле richTextBox користувач вводить текст, в поле textBox вводить склад. В 2-ге поле richTextBox виводяться всі слова з цим складом.

**Варіант 7**

Написати програму, у якій є поле richTextBox та 2 поля textBox. В richTextBox користувач вводить текст, в поля textBox – слово з тексту та слово для заміни. Заміна робиться по натисненні кнопки «Замінити».

**Варіант 8**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує найдовше слово у тексті та число його повторів у тексті.

**Варіант 9**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує спочатку всі окличні речення, потім питальні, потім всі інші.

**Варіант 7**

Написати програму, у якій є поле richTextBox та 2 поля textBox. В richTextBox вводиться текст, а в І поле textBox – слово для пошуку. В ІІ поле textBox виводиться скільки разів дане слово зустрічається в тексті.

**Варіант 8**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує всі речення в тексті у зворотному порядку.

**Варіант 9**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує введений текст міняючи у всіх словах першу і останню букви місцями.

**Варіант 10**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує речення з введеного тексту, в якому найчастіше зустрічається буква «О».

**Варіант 11**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує лише ті речення, в яких є цифри та кількість таких речень.

**Варіант 12**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує лише ті речення, які починаються на слово з однієї букви.

**Варіант 13**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує лише ті слова, які починаються і закінчуються на однакову літеру.

**Варіант 14**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує замість букв ­– їх номер в алфавіті, а замість цифр – їх назву.

**Варіант 15**

Написати програму, у якій є два текстових поля (richTextBox). В 1 текстове поле користувач вводить текст, у 2-ге програма записує лише ті слова, кількість літер в яких – просте число.

***Контрольні питання:***

1. Як ініціалізувати рядкову змінну?
2. Які ви знаєте методи для роботи з рядками?
3. За допомогою яких операторів можна змінити регістр літер у рядку?
4. Що таке кодування Unicode?
5. Які засоби роботи з окремими символами надає C#?
6. Яке основне обмеження має клас string?
7. Як визначити кількість символів у рядку?
8. Чим відрізняються класи String і StringBuilder?
9. Як об'єднати два об'єкти StringBuilder?
10. Чому іноді для роботи з рядками краще використовувати клас StringBuilder?