РЯДКИ - ОБ'ЄКТИ C

У C # рядки є об'єктами. Призначені для роботи з рядками символів в кодуванні Unicode, є вбудованим типом C #. Ключове слово типу - string. Йому відповідає базовий клас System.String бібліотеки .NET. Таким чином, string - це контрольний тип. Рядки типу string відносяться до так званим незмінним ти-пам даних.

Основні концепції

• Для подання рядком інформації в .NET існує клас System.String. У нього є псевдонім string, для якого в C # реалізована вбудована підтримка.

• Рядки задаються в подвійних лапках.

• Рядок може містити керуючі послідовності, які починаються зі зворотним похилій риси.

• Є можливість створювати дослівні (verbatim) рядки, де кожен символ сприймається як є. Такі рядки починаються з @

string s3 = @ "Text \ nText1 \ t \\";

• Як і всі в C # рядок - це об'єкт. І отже у об'єкту є властивості і методи. Слід зазначити, що рядок є незмінним об'єктом (immutable), тобто будь-які методи роботи з рядками повертають оброблені копії, а оригінал залишається незмінним.

• Індексація рядки йде від 0.

Оголошення рядків. Конструктори класу string

Об'єкти класу String оголошуються всі інші об'єкти простих типів - з явною або відкладеної инициализацией, з явним або неявним викликом конструктора класу. Найчастіше, при оголошенні строкової змінної конструктор явно не викликається, а ініціалізація задається строкової константою. Але у класу String досить багато конструкторів. Вони дозволяють сконструювати рядок з:

• символу, повтореного заданий число раз;

• масиву символів char [];

• частини масиву символів.

Розглянемо приклад ініціалізації рядків різними конструкторами.

```
private void button1_Click (object sender, EventArgs e)
    {
       // конструктори
       string world = "Мир";
       string sssss = new string ( 's', 5);
       char [] yes = "Yes" .ToCharArray ();
       string stryes = new string (yes);
       string strye = new string (yes, 0, 2);
       label1.Text = "world =" + world + "sssss =" + sssss + "stryes =" + stryes +
"strye =" + strye;

    }
```

Об'єкт world створений без явного виклику конструктора, а об'єкти sssss, stryes, strye створені різними конструкторами класу String.

Зауважте, не допускається явний виклик конструктора за замовчуванням - конструктора без параметрів. Немає також конструктора, якому в якості аргументу можна передати звичайну строкову константу. Відповідні оператори в тексті закоментовані.

Операції над рядками

Над рядками визначені наступні операції:

• присвоювання (=);

• дві операції перевірки еквівалентності (==) і (! =);

• конкатенація або зчеплення рядків (+);

• взяття індексу ([]).

Почну з привласнення, що має важливу особливість. Оскільки string - це контрольний тип, то в результаті привласнення створюється посилання на константну рядок, збережену в "купі". З однієї і тієї ж строкової константою в "купі" може бути пов'язане кілька змінних строкового типу. Але ці змінні не є псевдонімами - різними іменами одного і того ж об'єкта. Справа в тому, що рядкові константи в "купі" не змінюються, тому коли одна з змінних отримує нове значення, вона зв'язується з новим сталою об'єктом в "купі". Решта змінні зберігають свої зв'язки. Для програміста це означає, що семантика привласнення рядків аналогічна семантиці значимого присвоювання.

На відміну від інших довідкових типів, операції, перевіряючі еквівалентність, порівнюють значення рядків, а не посилання. Ці операції виконуються як над значущими типами.

Бінарна операція "+" зв'язує два рядки, приписуючи другий рядок до хвоста першої.

Можливість взяття індексу при роботі з рядками відображає той приємний факт, що рядок можна розглядати як масив і отримувати без праці кожен її символ. Кожен символ рядка має тип char, доступний тільки для читання, але не для запису.

Ось приклад, в якому над рядками виконуються дані операції:

Без констант не обійтися. У C # існують два види строкових констант:

• звичайні константи, які представляють рядок символів, укладену в лапки;

• @ -константи, задані звичайної константою с попереднім знаком @.

У звичайних константи деякі символи інтерпретуються особливим чином. Пов'язано це насамперед з тим, що необхідно вміти ставити в рядку недруковані символи, такі, як, наприклад, символ табуляції. Виникає необхідність ставити символи їх кодом - у вигляді escape-послідовностей. Для всіх цих цілей використовується комбінація символів, що починається

символом "\" - зворотна коса риска. Так, пари символів: "\ n", "\ t", "\\", "\" "задають відповідно символ переходу на новий рядок, символ табуляції, сам символ зворотної косої межі, символ лапки, що вставляється в рядок, але не сигналізує про її закінчення. Комбінація "\ xNNNN" задає символ, який визначається шістнадцятковим кодом NNNN. Хоча таке рішення виникаючих проблем цілком природно, іноді виникають неудо бства: наприклад, при завданні констант, що визначають шлях до файлу, доводиться кожен раз подвоювати символ зворотної косої межі. Це одна з причин, по якій з'явилися @ -константи.В @ -константи все символи трактуються в повній відповідності з їх зображенням. Тому шлях до файлу краще ставити @ константа. Єдина проблема в таких випадках: як поставити символ лапки, щоб він не сприймався як кінець самої константи. Рішенням є подвоєння символу. Незмінний клас stringВ мовою C # існує поняття незмінний (immutable) клас. Для такого класу неможливо змінити значення об'єкта при виклику його методів. Динамічні методи можуть створювати новий об'єкт, але не можуть змінити значення існуючого об'єкта.К таким незмінним класів відноситься і клас String. Жоден з методів цього класу не змінює значення існуючих об'єктів. Звичайно, деякі з методів створюють нові значення і повертають в якості результату нові рядки. Неможливість змінювати значення рядків стосується не тільки методів. Аналогічно, при роботі з рядком як з масивом дозволено тільки читання окремих символів, але не їх заміна. Оператор присвоювання в якому робиться спроба змінити перший символ рядка, неприпустимий, а тому закоментований .// Незмінні значення s1 = "Zenon"; ch1 = s1 [0]; // S1 [0] = 'L'; Статичні властивості і методи класу StringМетод ОпісаніеEmpty Повертається порожній рядок. Властивість зі статусом read only Compare Порівняння двох рядків. Метод перевантажений. Реалізації методу дозволяють порівнювати як рядки, так і підрядка. При цьому можна враховувати або не враховувати регістр, особливості національного форматування дат, чисел і т.д.CompareOrdinal Порівняння

двох рядків. Метод перевантажений. Реалізації методу дозволяють порівнювати як рядки, так і підрядка. Порівнюються коди сімволовConcat Конкатенация строк. Метод перевантажений, допускає зчеплення довільного числа строкСору Створюється копія строкіFormat Виконує форматування відповідно до заданих специфікацій формата.Intern, IsIntern Відшукується і повертається посилання на рядок, якщо така вже зберігається у внутрішньому пулі даних. Якщо ж рядка немає, то перший з методів додає рядок у внутрішній пул, другий - повертає null. Методи застосовуються зазвичай тоді, коли рядок створюється з використанням будівника рядків - класу StringBuilder Join Конкатенація масиву рядків в єдиний рядок. При конкатенації між елементами масиву вставляються роздільники. Операція, задана методом Join, є зворотною до операції, заданої методом Split. Останній є динамічним методом і, використовуючи роздільники, здійснює поділ рядка на елементиМетод FormatМетод Format визначає формат виведення рядка на екран. Як і більшість методів, є перевантаженим і може викликатися з різним числом параметрів. Перший необов'язковий параметр методу задає провайдера, що визначає національні особливості, які використовуються в процесі форматування. В якості такого параметра повинен бути заданий об'єкт, який реалізує інтерфейс System.IFormatProvider. Якщо цей параметр не заданий, то використовується культура, задана за замовчуванням. Ось приклади двох реалізацій цього методу: public static string Format (string, object); public static string Format (IFormatProvider, string, params object []); Параметр типу string задає форматується рядок. Заданий рядок містить один або кілька форматів, вони розпізнаються за рахунок оточуючих формат фігурних дужок. Число форматів, вставлених в рядок, визначає і число об'єктів, що передаються при виклику методу Format. Кожен формат визначає форматування об'єкта, на який він посилається і який, після перетворення його в рядок, буде підставлена у результуючий рядок замість формату. Метод Format як результат повертає передану йому рядок, де всі

специфікації формату замінені рядками, отриманими в результаті форматування об'ектов.Общій синтаксис, специфікує формат, такий: {N [, M [: <коди_форматірованія>]]} Обов'язковий параметр N задає індекс об'єкта , що заміняє формат. Можна вважати, що методу завжди передається масив об'єктів, навіть якщо фактично переданий один об'єкт. Індексація об'єктів починається з нуля, як це прийнято в масивах. Другий параметр M, якщо він заданий, визначає мінімальну ширину поля, яке відводиться рядку, що вставляється замість формату. Третій необов'язковий параметр задає коди форматування, що вказують, як слід форматувати об'єкт. Наприклад, код C (Currency) говорить про те, що параметр повинен форматироваться як валюта з урахуванням національних особливостей подання. Код P (Percent) задає форматування у вигляді відсотків з точністю до сотої частки. Спочатку демонструється, що і явний, і неявний виклики методу Format дають один і той же результат. У подальших прикладах показано використання різних специфікацій формату з різними числом параметрів і різними кодами форматування. Зокрема, показаний висновок відсотків і валют. В останньому прикладі з валютами дем онстріруется завдання провайдером національних особливостей. З цією метою створюється об'єкт класу CultureInfo, ініціалізований так, щоб він ставив особливості форматування, прийняті в США. Зауважте, клас CultureInfo успадковує інтерфейс IFormatProvider. Російські національні особливості форматування встановлені за замовчуванням. При необхідності їх можна встановити таким же чином, як це зроблено для США, задавши відповідно константу "ru-RU" .Методи Join і SplitМетоди Join і Split виконують над рядком тексту взаємно зворотні перетворення. Динамічний метод Split дозволяє здійснити розбір тексту на елементи. Статичний метод Join виконує зворотну операцію, збираючи рядок з елементов.Заданний рядком текст часто являє собою сукупність структурованих елементів - абзаців, пропозицій, слів, дужкових виразів і т.д. При роботі з таким текстом необхідно розділити його на елементи,

користуючись спеціальними роздільниками елементів, - це можуть бути прогалини, дужки, знаки пунктуації. Практично подібні завдання виникають постійно при роботі зі структурованими текстами. Методи Split і Join полегшують рішення цих задач.Дінаміческій метод Split, як зазвичай, перевантажений. Найбільш часто використовувана реалізація має наступний синтаксис: public string [] Split (params char []) На вхід методу Split передається один або кілька символів, інтерпретованих як роздільники. Об'єкт string, що викликав метод, розділяється на підрядка, обмежені цими роздільниками. З цих подстрок створюється масив, що повертається в якості результату методу. Інша реалізація дозволяє обмежити число елементів, що повертається массіва.Сінтаксіс статичного методу Join такий: public static string Join (string delimiters, string [] items) Як результат метод повертає рядок, отриману конкатенацией елементів масиву items, між якими вставляється рядок роздільників delimiters. Як правило, рядок delimiters складається з одного символу, який і розділяє в результуючому рядку елементи масиву items; але в окремих випадках обмежувачем може бути рядок з декількох сімволов.Прімер. Необхідно розбити рядок - складнопідрядне речення на прості речення. після чого розділити пропозиції на слова, а потім зібрати текст назад. Зверніть увагу, що методи Split і Join добре працюють, коли при розборі використовується тільки один роздільник. В цьому випадку збірка дійсно є зворотною операцією і дозволяє відновити вихідну рядок. Якщо ж при розборі задається деякий безліч роздільників, то виникають дві проблеми: • неможливо при складанні відновити рядок в колишньому вигляді, оскільки не зберігається інформація про те, який з роздільників був використаний при розборі рядка. Тому при складанні між елементами вставляється один роздільник, можливо, що складається з декількох символів; • при розборі двох поспіль роздільників передбачається, що між ними знаходиться порожнє слово. Зверніть увагу в тексті нашого прикладу, як і належить, після коми слід пробіл. При розборі тексту на слова як

роздільники вказані символи пробілу і коми. З цієї причини в масиві слів, отриманому в результаті розбору, є порожні слова.Еслі при розборі пропозиції на слова використовувати як роздільник тільки пробіл, то порожні слова не з'являться, але кома буде частиною деяких слов.Как завжди, є кілька способів впоратися з проблемою. Один з них полягає в тому, щоб написати власну реалізацію цих функцій, інший - в коригуванні отриманих результатів, третій - у використанні більш потужного апарату регулярних виразів, і про нього ми поговоримо трохи позже.Дінаміческіе методи класу StringОпераціі, дозволені над рядками в C #, різноманітні. Методи цього класу дозволяють виконувати вставку, видалення, заміну, пошук входження підрядка в рядок. Клас String успадковує методи класу Object, частково їх перевизначаючи. Клас String успадковує і, отже, реалізує методи чотирьох інтерфейсів: IComparable, ICloneable, IConvertible, IEnumerable. Три з них вже розглядалися при описі класів-массівов.Рассмотрім найбільш характерні методи при роботі зі строкамі.Сводка методів, наведена в таблиці, дає досить повну картину широких можливостей, наявних при роботі з рядками в C #. Слід пам'ятати, що клас string є незмінним. Тому Replace, Insert і інші методи являють собою функції, які повертають новий рядок в якості результату і не змінюють рядок, що викликала метод.Метод ОпісаніеInsert Вставляє підрядок в задану позіціюRemove Видаляє подстроку в заданій позіціїReplace Замінює подстроку в заданій позиції на нову подстрокуSubstring Виділяє подстроку в заданій позіціїIndexOf , IndexOfAny, LastIndexOf, LastIndexOfAny Визначаються індекси першого і останнього входження заданого підрядка або будь-якого символу з заданого набораStartsWith, EndsWith Повертається true або false, залежно від того, починає ся або закінчується рядок заданої подстрокойPadLeft, PadRight Виконує набивання потрібним числом прогалин на початку і в кінці строкіTrim, TrimStart, TrimEnd Зворотні операції до методів Pad. Видаляються прогалини на початку і в кінці рядка, або тільки з одного її концаToCharArray

Перетворення рядка в масив сімволовКласс StringBuilder - будівник строкКласс string не дозволяє змінювати існуючі об'єкти. Строковий клас StringBuilder дозволяє компенсувати цей недолік. Цей клас належить до змінним класам і його можна знайти в просторі імен System.Text. Розглянемо клас StringBuilder подробнее.Об'явленіе рядків. Конструктори класу StringBuilderОб'екти цього класу оголошуються з явним викликом конструктора класу. Оскільки спеціальних констант цього типу не існує, то виклик конструктора для ініціалізації об'єкта просто необхідний. Конструктор класу перевантажений, і поряд з конструктором без параметрів, що створює порожній рядок, є набір конструкторів, яким можна передати дві групи параметрів. Перша група дозволяє задати рядок або подстроку, значенням якої буде инициализироваться створюваний об'єкт класу StringBuilder. Друга група параметрів дозволяє задати ємність об'єкта - обсяг пам'яті, що відводиться даному екземпляру класу StringBuilder. Кожна з цих груп не є обов'язковою і може бути опущена. Прикладом може служити конструктор без параметрів, який створює об'єкт, ініціалізований символом нового рядка, і з деякою ємністю, заданої за замовчуванням, значення якої залежить від реалізації. Наведу як приклад синтаксис трьох конструкторів: • public StringBuilder (string str, int cap). Параметр str задає рядок ініціалізації, cap - ємність об'єкта; • public StringBuilder (int curcap, int maxcap). Параметри curcap і maxcap задають початкову і максимальну ємність об'єкта; • public StringBuilder (string str, int start, int len, int cap). Параметри str, start, len задають рядок ініціалізації, cap - ємність об'єкта.Операції над строкамиНад рядками цього класу визначені практично ті ж операції з тієї ж семантикою, що і над рядками класу String: • присвоєння (=); • дві операції перевірки еквівалентності (==) і (! =); • взяття індексу ([]) .Операція конкатенації (+) не визначена над рядками класу StringBuilder, її роль грає метод Append, дописувати новий рядок в хвіст вже существующей.Со рядком цього класу можна працювати як з масивом, але, на відміну від класу String, тут вже все

робиться як треба: допускається не тільки читання окремого символу, а й його зміна. Цей приклад демонструє можливість виконання над рядками класу StringBuilder тих же операцій, що і над рядками класу String. В результаті привласнення створюється додаткова посилання на об'єкт, операції перевірки на еквівалентність працюють зі значеннями рядків, а не з посиланнями на них. Конкатенацію можна замінити викликом методу Append. З'являється нова можливість - змінювати окремі символи строкі.Основние методиУ класу StringBuilder методів значно менше, ніж у класу String. Це і зрозуміло - клас створювався з метою дати можливість змінювати значення рядка. З цієї причини у класу є основні методи, що дозволяють виконувати такі операції над рядком як вставка, видалення і заміна подстрок, але немає методів, подібних пошуку входження, які можна виконувати над звичайними рядками. Технологія роботи зазвичай така: конструюється рядок класу StringBuilder; виконуються операції, що вимагають зміна значення; отримана рядок перетвориться в рядок класу String; над цим рядком виконуються операції, які не потребують зміни значення рядка. Давайте трохи детальніше розглянемо основні методи класу StringBuilder: • public StringBuilder Append (<об'єкт>). До рядку, що викликала метод, приєднується рядок, отримана з об'єкта, який переданий методу в якості параметра. Метод перевантажений і може приймати на вході об'єкти всіх простих типів, починаючи від char і bool до string і long. Оскільки об'єкти всіх цих типів мають метод ToString, завжди є можливість перетворити об'єкт в рядок, яка і приєднується до заданій стрічці. Як результат повертається посилання на об'єкт, що викликав метод. Оскільки возвращаемую посилання нічому привласнювати не потрібно, то правильніше вважати, що метод змінює значення рядка; • public StringBuilder Insert (int location, <об'єкт>). Метод вставляє рядок, отриману з об'єкта, в позицію, зазначену параметром location. Метод Append є окремим випадком методу Insert; • public StringBuilder Remove (int start, int len). Метод видаляє

підрядок довжини len, що починається з позиції start; • public StringBuilder Replace (string str1, string str2). Всі входження підрядка str1 замінюються на рядок str2; • public StringBuilder AppendFormat (<рядок форматів »,« об'єкти>). Метод є комбінацією методу Format класу String і методу Append. Рядок форматів, передана методу, містить тільки специфікації форматів. У соответс твии з цими специфікаціями знаходяться і форматируются об'єкти. Отримані в результаті форматування рядка приєднуються в кінець вихідної строкі.За винятком методу Remove, всі розглянуті методи є перевантаженими. В їх описі дана схема виклику методу, а не точний синтаксис перевантажених реалізацій. Наведу приклади, щоб продемонструвати, як викликаються і як працюють ці методи: В цьому фрагменті коду конструюються два рядки. Перша з них створюється з рядків і булевих значень true і false. Для конструювання використовуються методи Insert і Append. Другий рядок конструюється в циклі з застосуванням методу AppendFormat. Результатом цього конструювання є рядок, в якій прості речення вихідного тексту пронумеровани.Обратіте увагу, що сконструйована другий рядок передається в звичайну рядок класу String. Ніяких проблем перетворення рядків одного класу в інший клас не виникає, оскільки всі об'єкти, в тому числі, об'єкти класу StringBuilder, мають за визначенням методом ToString.