11.Strings

Работа със символи и символни низове в .NET

Бранислав Абаджимаринов

Съдържание

- Основни понятия
- Unicode
- Символен тип
- Символен низ
- Работа с низове
- Форматиране на низове
- Култури в .NET
- Ескейпинг последователности
- Производителност и оптимизации

Основни понятия

- Символна кодова таблица и кодова схема (character table & encoding)
- Популярни кодови схеми
  - ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) (ISO 646)
    - 128 символа – 95 изобразими, 33 контролни
    - 7 bit за символ
    - само латински сиволи
    - Често се допълва до 8 bit с избрана азбука
    - Неподдържаните символи се заменят с "?"

Основни понятия

- Популярни кодови схеми
  - ISO 8859
    - 8 bit за символ
    - състои се от 15 части – ISO 8859-1 до 8859-15
  - MS-Windows character sets
    - 8 bit
    - Регионални кодови таблици
    - Windows 1251 & Windows 1252

Unicode

- Стандарт, който съпоставя числов код на символ
- Унифицирана кодова таблица
- Глобално номериране на символите
- Поддържа всички езици
- 107,000 уникални символа
- Поддържа съставни символи
- Unicode Consortium – www.unicode.org

Unicode Кодови схеми

- UTF-8
  - 1,2,3 или 4 байта на символ
  - Латинските символи заемат 1 байт
  - Запазва номерата от ASCII
- UTF-16
  - 2 или 4 байта на символ
- UTF-32
  - Фиксирана големина от 4 байта на символ

Символен тип

- Символните в .NET се представят от типа **`System.Char`**
- Съкратено наименование **`char`**

```
char symbol = 'A';
```

- Стойностен тип
- Използва UTF-16 кодираща схема
- Една графема може да се съдържа в повече от един **`char`**

Символен тип - Методи

- Имплементирани са като статични методи на типа Char
- За класификация
  - **IsControl()**
  - **IsDigit()**
  - **IsLetter()**
  - **IsLetterOrDigit()**
  - **IsNumber()**
  - **IsPunctuation()**
  - **IsSeparator()**
  - **IsSymbol()**
  - **Char.IsWhiteSpace()**
  - **GetUnicodeCategory()** – връща енумерирания тип **System.Globalization.UnicodeCategory**

Символен тип - Методи

- За манипулиране на главни и малки букви
  - **`IsLower()`**
  - **`IsUpper()`**
  - **`ToLower()`**
  - **`ToUpper()`**

Символен низ

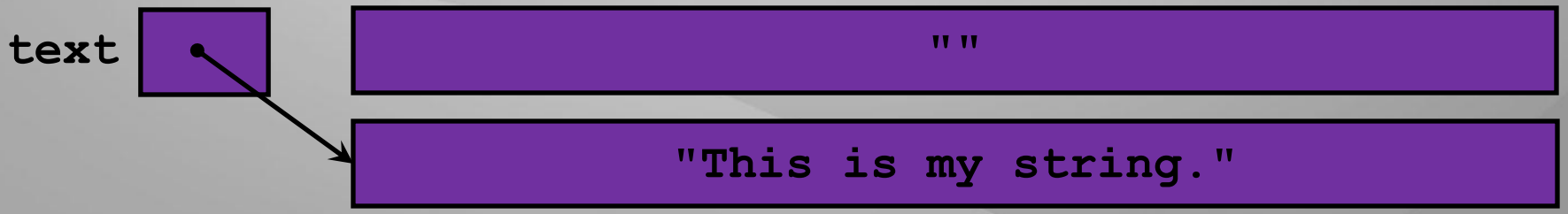
- Символен низ в .NET се представя с типът **`System.String`**
- Съкратено име **`string`**
- Неизменима последователност от символи от тип **`System.Char`**
- Използва Unicode стандарта
- Референтен тип

Особености на типа

```
static void Main()
{
    string text = "";
    text += "This is my string.";
    text += "My string is amazing!";

    Console.WriteLine("text = {0}", text);
}
```

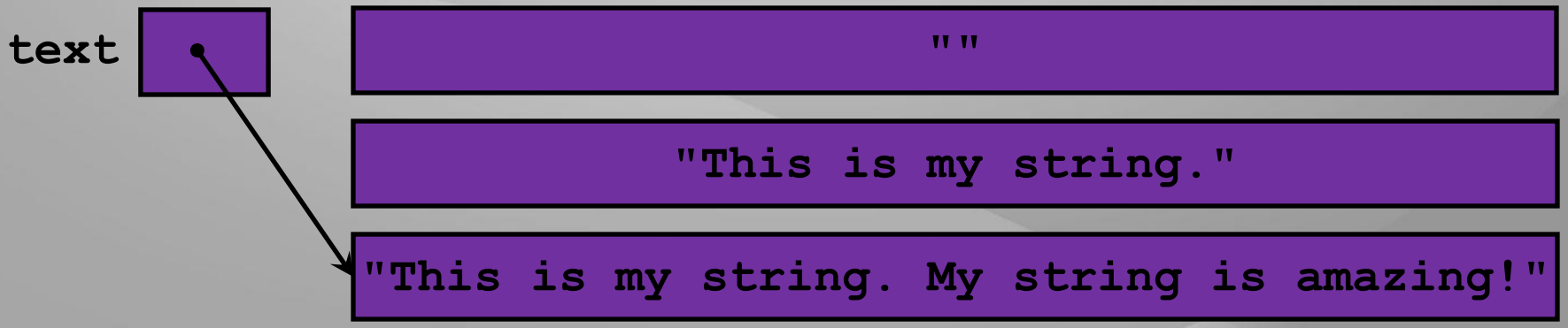
Особености на типа

```
static void Main()
{
    string text = "";
    text += "This is my string.";
    text += "My string is amazing!";

    Console.WriteLine("text = {0}", text);
}
```



Особености на типа

```
static void Main()
{
    string text = "";
    text += "This is my string.";
    text += "My string is amazing!";

    Console.WriteLine("text = {0}", text);
}
```



Работа с низове

- Инстанциране на низ

```
string text = "This is my string. My string is amazing!"
```

- ToString() – преобразува обект в низ
- Length – връща броя на char-ове в низа (може да не съвпада с броя видими символи)
- == и != - Оператори за сравнение на стойностите на два низа

Работа с низове

- Методи **`Equals(string)`** и **`String.Compare(string, string)`** – сравнение с опции за малки и главни букви
- + и += - оператор за конкатенация и за добавяне на надставка
- Индексатор за достъп до символи

```
string courseName = "Programming .NET & WPF";
char and = courseName[17];
```

Работа с низове

- **`IndexOf(string)`** – връща индекса на първото срещане на даден низ в друг низ. Ако не се среща връща -1
- **`LastIndexOf(string)`** – връща индекса на последно срещане на зададения низ
- И двата метода имат множество предефиниции

Работа с низове

- **`Substring(int startIndex, int Length)`** – връща подниз започващ от зададения индекс с дължина зададено число
- **`StartsWith(string)`**- проверява дали низът започва с подадената стойност
- **`EndsWith(string)`**- проверява дали низът завършва с подадената стойност

Работа с низове

- **`Split(params char[])`**

  - разделя даден низ на низове по подаден списък от разделители
- **`Join(string, string[])`**

  - съединява масив от низове, като между всеки два поставя зададен низ
- **`ToLower()`** и **`ToUpper()`**

  - връща нова инстанция на низа в която всички букви са малки/главни

Работа с низове

- **`PadLeft(int, char)`, `PadRight(int, char)`** – допълва (от ляво или дясно) низ до зададена дължина със зададен символ
- **`Trim(params char[])`, `TrimStart`, `TrimEnd`** – премахва зададени символи от началото и края на низ
- Правило : Потребителските имена се trim-ват, паролите - никога

Работа с низове

- **`Insert(int index, string)`**
  - връща низ получен от първоначалния чрез добавяне на подниз на позиция започваща от index
- **`Remove(int index, int count)`**
  - връща низ в който е премахнат даден подниз започващ от index и с дължина count
- **`Replace(string oldVal, string newVal)`**
  - връща низ, в който всяко срещане на даден подниз е заменено с нова зададена стойност

Форматиране на низове

- Format(string, params object[])
  - връща низ в който всички форматиращи символи са заменени със съответните им стойности по ред на подаване

```
String.Format("This is my {0}. My {0} is {1}!",
objectName, value);
```

- ToString() също приема форматиращи низове
- Видове форматиращи низове

Култури

- Съдържат информация за
  - Формат на дати и време
  - Формат на цифри и валута
  - Смяна на малка и главна буква
  - Сравняване на низове (от дясно на ляво или от ляво на дясно)
  - Календар и други

Култури

- Класът **`System.Globalization.CultureInfo`** се използва за получаване на информация за дадена култура
- По подразбиране се използва културата на текущата нишка на приложението

Култури

- Взимане на инстанция на CultureInfo

```
CultureInfo info = new CultureInfo("bg-BG");
```

  - Език - bg
  - Страна – BG
- Взимане на всички култури

```
CultureInfo.GetCultures(CultureTypes.AllCultures);
```

- Култура при форматиране

```
String.Format(new CultureInfo("en-US"), "{0:C}", 100);
```

Parse & TryParse

- За преобразуване на текст в обект от конкретен тип се използват статичните методи Parse & TryParse на конкретния тип
- Parse(string)
  - връща инстанция на преобразувания обект
  - ако не може да преобразува низа в конкретния тип обект хвърля изключение

```
int five = int.Parse("5");
```

Parse & TryParse

- TryParse(string, out variable)
  - Връща булева стойност
  - true ако може да преобразува низа до обект от указания тип, иначе false
  - Не хвърля изключение
  - Връща получения обект като out параметър
  - За предпочитане в повечето случаи

```
int five;
int.TryParse("5", out five);
```

Ескейпинг последователности

- Символ или последователност от символи със специално значение
- При компилация на кода се преобразуват в символите, които заместват

Ескейпинг последователности

- \xXX – представлява ASCII кодиран символ
- \uXXXX – представлява Unicode символ
  - X е шеснадесетична стойност с главни букви и цифри
- \n – нов ред (line feed) (\x0A)
- \r – символа carriage return (CR) (\x0D)
- \" – двойни кавички в низ (\x22)
- \' – единични кавички в низ (\x27)
- \t – символа табулация (\x09)
- \\ – обратно наклонена черта (\x5C)
- \0 – символ null за край на стринг (\x00)
  - Не се ползва явно. Слага се автоматично с цел съвместимост с други среди и езици за програмиране

Цитиран низ

- Започва със символа @ и кавичка
- Завършва с кавичка – "
- Взима буквално всичко между двете кавички - нов ред, интервал, \
- За кавичка се използва двойна – ""
- Използва се често за пътища на файлова система и регулярни изрази

```
string path = @" C:\Windows\System32\drivers\etc\hosts";
```

Производителност

- Всяка промяна в символен низ води до създаване на нова инстанция на референтния тип string
- Инстанцирането на нов обект в .NET е лесен процес
- Изчистването на референтрните типове обаче е тежък и скъп процес
- При честа конкатенация на низове се използва **`StringBuilder`**

StringBuilder

- Използва се при нужда от честа манипулация на стойността на низ
- Заделя буферна памет и не създава нова инстанция при промяна
- При надхвърляне на заделената памет се заделя още
- За да получим обикновен низ от **`StringBuilder`** обект викаме метода **`ToString()`**

StringBuilder

```
StringBuilder resultBuilder = new StringBuilder();
for (int i = 0; i < 10000; i++)
{
    resultBuilder.Append(i.ToString());
}
string result = resultBuilder.ToString();
```

VS

```
string result = String.Empty;
for (int i = 0; i < 10000; i++)
{
      result += i.ToString();
}
```

StringBuilder

- Дублира методите на String, но не създава нови инстанции, а променя текущата
  - **`Append(…)`**
  - **`AppendLine(...)`**
  - **`AppendFormat(…)`**
  - **`Insert(…)`**
  - **`Remove(…)`**
  - **`Replace(…)`**
- **`Capacity`** – показва поддържания буфер

Интернирани низове

- Всяко .NET приложение поддържа таблица на интернираните стрингове – intern pool
- Специално място в паметта, тип хеш таблица от низове
- Ако два string обекта са интернирани и имат една и съща стойност – то те сочат към един и същи низ в intern pool-а

Интернирани низове

- Сравняването на интернирани стрингове е много бързо – сравняват се техните референции
- CLR автоматично интернира всички символни литерали в програмата
- Възможно е изрично интерниране на низове с цел пестене на памет

Интернирани низове

- **`String.Intern(string)`** - добавя низа в intern pool-а и връща референция към обекта
- **`String.IsInterned(string)`**- проверява дали даден низ е интерниран и ако е връща референция към низ-а в intern pool-а

Въпроси?

Задачи

- Да се напише метод със сигнатура CutText(string text, int length, string suffix), който връща подниз на text с дължина length, започващ от началото и долепя в края на новия низ стойността на suffix
- Напишете метод, който по подаден свободен текст връща списък на всички думи в текста и брой на тяхното срещане

Задачи

- Напишете метод, който по подаден текст съдържащ отварящи и затварящи скоби премахва текста затворен в скоби заедно със самите скоби. Методът трябва да се справя и с вложени скоби. Например в "Am I writing ((my) homework (or what))?" изходът да е "Am I writing ?"