

Сравнение строк. Ординальное сравнение или сравнение, чувствительное к культуре

При сравнении строк используются два базовых алгоритма: алгоритм ординального сравнения и алгоритм сравнения, чувствительного к культуре.

В случае ординального сравнения символы интерпретируются как числа (согласно числовым кодам Unicode).

В случае сравнения, чувствительного к культуре, символы интерпретируются со ссылкой на конкретный словарь.

Существуют две специальные культуры: “текущая культура”, которая основана на настройках, получаемых из панели управления компьютера, и “инвариантная культура”, которая является одной и той же на всех компьютерах (и полностью соответствует американской культуре).

Для сравнения эквивалентности удобны оба алгоритма.

Сравнение строк

При упорядочивании сравнение, чувствительное к культуре, почти всегда предпочтительнее: для алфавитного упорядочения строк необходим алфавит. Ординальное сравнение полагается на числовые коды Unicode, которые выстраивают английские символы в алфавитном порядке — но не в точности так, как можно было бы ожидать.

Например, предполагая чувствительность к регистру, рассмотрим строки "Atom", "atom" и "Zoom". В случае инвариантной культуры они располагаются в следующем порядке:

"Atom", "atom", "Zoom"

При ординальном сравнении результат выглядит так:

"Atom", "Zoom", "atom"

Инвариантная культура инкапсулирует алфавит, в котором символы в верхнем регистре находятся рядом со своими двойниками в нижнем регистре (AaBbCcDd...). При ординальном сравнении сначала идут все символы в верхнем регистре, а затем — все символы в нижнем регистре (A...Z, a...z) (производится "возврат" к набору символов ASCII, появившемуся

Сравнение эквивалентности строк

Операция `==` в типе `string` выполняет ординальное сравнение, чувствительное к регистру.

Метода `string.Equals` в случае вызова без параметров работает аналогично.

```
public bool Equals(string value, StringComparison comparisonType);  
public static bool Equals (string a, string b, StringComparison comparisonType);  
public enum StringComparison  
{  
    CurrentCulture,          // Чувствительное к регистру  
        CurrentCultureIgnoreCase,  
    InvariantCulture,       // Чувствительное к регистру  
        InvariantCultureIgnoreCase,  
    Ordinal,                // Чувствительное к регистру  
        OrdinalIgnoreCase  
}
```

```
using System.Globalization;
...
static void Main()
    {   CultureInfo current = CultureInfo.CurrentCulture;
        Console.WriteLine("The current culture is {0}",
current.Name);
        CultureInfo newCulture;
        if (current.Name.Equals("ru-RU"))
            newCulture = new CultureInfo("en-US");
        else
            newCulture = new CultureInfo("fr-FR");

        CultureInfo.CurrentCulture = newCulture;
        Console.WriteLine("The current culture is now {0}",
            CultureInfo.CurrentCulture.Name);
    }
```

<https://habr.com/ru/company/enterra/blog/237209/>

Сравнение порядка строк

Метод экземпляра `CompareTo` класса `String` выполняет чувствительное к культуре и регистру сравнение порядка. В отличие от операции `==` метод `CompareTo` не использует ординальное сравнение: для упорядочивания намного более полезен алгоритм сравнения, чувствительного к культуре.

```
public int CompareTo (string strB);
```

Для других видов сравнения можно вызывать статические методы:

```
public static int Compare (string strA, string strB,  
                           StringComparison comparisonType);
```

```
public static int Compare (string strA, string strB, bool ignoreCase,  
                           CultureInfo culture);
```

```
public static int Compare (string strA, string strB, bool ignoreCase);
```

```
public static int CompareOrdinal (string strA, string strB);
```

Все методы сравнения порядка возвращают положительное,

Некоторые методы работы со строками

static int Compare(string strA, string strB, StringComparison comparisonType)

Возвращает отрицательное значение, если строка strA меньше строки strB ;

положительное значение, если строка strA больше строки strB ;

ноль, если сравниваемые строки равны. Способ сравнения определяется аргументом comparisonType

bool Equals(string value, StringComparison comparisonType)

Возвращает логическое значение true, если вызывающая строка имеет такое же значение, как и у аргумента value. Способ сравнения определяется аргументом comparisonType

int IndexOf(char value)

Осуществляет поиск в вызывающей строке первого вхождения символа, определяемого аргументом value. Возвращает индекс первого совпадения с искомым символом или -1, если он не

Язык программирования C#

int IndexOf(string value, StringComparison comparisonType)

Осуществляет поиск в вызывающей строке первого вхождения подстроки, определяемой аргументом value. Возвращает индекс первого совпадения с искомой подстрокой или -1, если она не обнаружена. Способ поиска определяется аргументом comparisonType

int LastIndexOf(char value)

Осуществляет поиск в вызывающей строке последнего вхождения символа, определяемого аргументом value. Возвращает индекс последнего совпадения с искомым символом или -1, если он не обнаружен

int LastIndexOf(string value, StringComparison comparisonType)

Осуществляет поиск в вызывающей строке последнего вхождения подстроки, определяемой аргументом value. Возвращает индекс последнего совпадения с искомой подстрокой или -1, если она не

Язык программирования C#

string ToLower(CultureInfo.CurrentCulture culture)

Возвращает вариант вызывающей строки в нижнем регистре. Способ преобразования определяется аргументом culture

string ToUpper(CultureInfo.CurrentCulture culture)

Возвращает вариант вызывающей строки в верхнем регистре. Способ преобразования определяется аргументом culture

string Substring(int индекс_начала, int длина)

возвращает подстроку вызывающей строки. индекс_начала обозначает начальный индекс в исходной строке, а длина — длину выбираемой подстроки.

Язык программирования C#

```
string str1 = "Пример работы со строками.";
string str2 = "Пример работы со строками.";
string str3 = "Строки в C#.";
```

```
string strUp, strLow; int result, idx;
```

```
Console.WriteLine("str1: " + str1);
Console.WriteLine("Длина строки str1: " + str1.Length);
```

```
strLow = str1.ToLower(CultureInfo.CurrentCulture); // Строчные буквы
strUp = str1.ToUpper (CultureInfo.CurrentCulture); // Прописные буквы
```

```
Console.WriteLine("Вывод строки str1 посимвольно.");
for (int i=0; i < str1.Length; i++)
Console.Write(str1[i]);
```

Язык программирования C#

```
string str1 = "Пример работы со строками.";
string str2 = "Пример работы со строками.";
string str3 = "Строки в C#.";

if (str1 == str2) Console.WriteLine("равны");

result = string.Compare(str1, str3, StringComparison.CurrentCulture);
if(result < 0) Console.WriteLine("Строка str1 меньше строки str3");

str2 = "Один Два Три Один";

idx = str2.IndexOf("Один", StringComparison.Ordinal);
// по порядковому номеру с учетом регистра
Console.WriteLine("Индекс первого вхождения: " + idx); // 0

idx = str2.LastIndexOf("Один", StringComparison.Ordinal);
Console.WriteLine("Индекс последнего вхождения: " + idx) ; // 13
```

Сравнение строк

При сравнении двух значений в .NET Framework проводится различие между концепциями сравнения эквивалентности и сравнения порядка.

Сравнение эквивалентности проверяет, являются ли два экземпляра семантически одинаковыми;

сравнение порядка выясняет, какой из двух экземпляров (если есть) будет следовать первым в случае расположения их по возрастанию или убыванию.

Для сравнения эквивалентности строк можно использовать операцию `==` или один из методов `Equals` типа `string`.

Для сравнения порядка строк можно применять либо метод экземпляра `CompareTo`, либо статические методы `Compare` и `CompareOrdinal`.

Язык программирования C#

Содержимое объекта типа `string` не подлежит изменению.

Если требуется строка в качестве разновидности уже имеющейся строки,

то следует создать новую строку, содержащую все необходимые изменения.

Неиспользуемые строковые объекты автоматически собираются сборщиком мусора

Переменные ссылки на строки (объекты типа `string`) могут быть изменены, следовательно могут ссылаться на другой объект.

Иногда полезно иметь возможность видоизменять строки.

Для этой цели в C# имеется класс `StringBuilder`, который определен в пространстве имен `System.Text` (*using System.Text;*). Он позволяет создавать строковые объекты, которые можно изменять.

Массивы строк

```
string[] str = { "Это", "простой", "тест." };
```

```
Console.WriteLine("Исходный массив: ");
```

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

```
Console.Write(str[i] + " ");
```

```
Console.WriteLine();
```

```
str[1] = "тоже";
```

```
Console.WriteLine("Видоизмененный массив: ");
```

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

```
Console.Write(str[i] + " ");
```

Применение строк в операторах switch

```
string[] strs = { "один", "два", "три", "два", "один" };
foreach (string s in strs)
{
    switch (s)
    {
        case "один":
            Console.WriteLine(1);
            break;
        case "два":
            Console.WriteLine(2);
            break;
        case "три":
            Console.WriteLine(3);
            break;
    }
}
```