Практическое занятие

По Процедурному программированию

Тема: Работа со строками

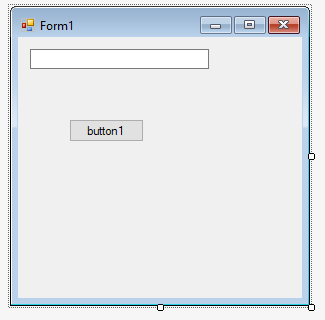
Строка — это последовательность символов. Вопреки распространенному мнению, строка НЕ является массивом символов, хотя общая организация и принципы хранения в памяти для строк и массивов схожи. Строки необходимы для организации взаимодействия с пользователем и реализации обработки текстовых данных.

Изначально для обработки текстовых данных под одно знакоместо отводилось 5 бит (32 возможных варианта). Этого было достаточно для хранения и передачи латиницы и некоторого количества служебных символов. Прописные и строчные буквы в таком кодировании не различались. Позднее под один символ уже стали выделять 7 бит (128 возможных вариантов). Этого достаточно для представления прописных и строчных символов латинской азбуки, цифр, знаков математических операций, пунктуации, служебных и некоторых других. Но этого недостаточно для одновременного представления латинских букв и символов алфавитов, не основанных на латинице (например, кириллицы). Поэтому размер одного символа был расширен до логичного значения в 8 бит (256 возможных символов). Этого хватает для одновременной работы с символами латиницы и какого-либо простого национального алфавита (в нашем случае кириллицы). Однако этого очевидно недостаточно для трехязычных текстов или же языков с иероглифическим письмом. Поэтому был составлен стандарт универсального кодирования символов — Unicode.

В C# для хранения символьных и строковых данных используется кодировка UTF-16, требующая хранение 2 байт информации на одно знакоместо. В соответствии со стандартом Unicode, одно знакоместо НЕ ОБЯЗАНО соответствовать одной букве. Одна буква может быть представлена несколькими подряд идущими знакоместами.

Давайте проверим это и одновременно изучим другие операции со строками.

Создадим приложение примерно такого вида:



В обработчике нажатия на кнопку button1 напишем код:

String s1 = "ё";

String s2 = "е\x308";

MessageBox.Show(String.Format("{0} -> {1}\n{2} -> {3}", s1, s1.Length, s2, s2.Length));

Запустим и убедимся, что хотя строки имеют разную длину (первая 1, вторая 2) они обе представляют один и тот же символ ё.

Для начала, решим какую-нибудь практическую задачу. Например, подсчитаем количество слов в строке. Введем простое определение, что слово — непрерывная последовательность букв. Это не соответствует фактическим границам слов, но в качестве грубого приближения пригодно к использованию. Соответственно разбирать будем текст, введенный пользователем в textBox1. Таким образом у нас получится примерно такой код:

String s1 = textBox1.Text;

int count = 0;

int wc = 0;

for (int i = 0; i < s1.Length; i++)

{

if ((s1[i] >= 'a' && s1[i] <= 'z') || ((s1[i] >= 'A') && (s1[i] <= 'Z')))

wc++;

else if (wc > 0)

{

wc = 0;

count++;

}

}

MessageBox.Show(count.ToString());

В данном случае мы рассматриваем только слова, составленные из латинских букв. Давайте добавим обработку кириллицы:

String s1 = textBox1.Text;

int count = 0;

int wc = 0;

for (int i = 0; i < s1.Length; i++)

{

if ((s1[i] >= 'a' && s1[i] <= 'z') || ((s1[i] >= 'A') && (s1[i] <= 'Z')) ||

(s1[i] >= 'а' && s1[i] <= 'я') || ((s1[i] >= 'А') && (s1[i] <= 'Я')))

wc++;

else if (wc > 0)

{

wc = 0;

count++;

}

}

MessageBox.Show(count.ToString());

Теперь проверим эту программу на следующем наборе тестов:

Я пришла!

Мама мыла раму.

The quick brown fox jumps over the lazy dog.

Съешь ещё этих мягких французских булок, да выпей же чаю.

Данные тесты должны пройти корректно.

Проверим еще на нескольких вариантах:

сторублируйте

да пребудет с тобой сила

It’s hammer time

В этих тестовых случаях получаемый результат должен быть на 1 меньше корректного. Почему это происходит? Просто мы не учитываем последнее слово в строке, если после него нет знака препинания (не буквы). Это можно легко исправить добавив всего пару строк:

String s1 = textBox1.Text;

int count = 0;

int wc = 0;

for (int i = 0; i < s1.Length; i++)

{

if ((s1[i] >= 'a' && s1[i] <= 'z') || ((s1[i] >= 'A') && (s1[i] <= 'Z')) ||

(s1[i] >= 'а' && s1[i] <= 'я') || ((s1[i] >= 'А') && (s1[i] <= 'Я')))

wc++;

else if (wc > 0)

{

wc = 0;

count++;

}

}

if (wc > 0)

count++;

MessageBox.Show(count.ToString());

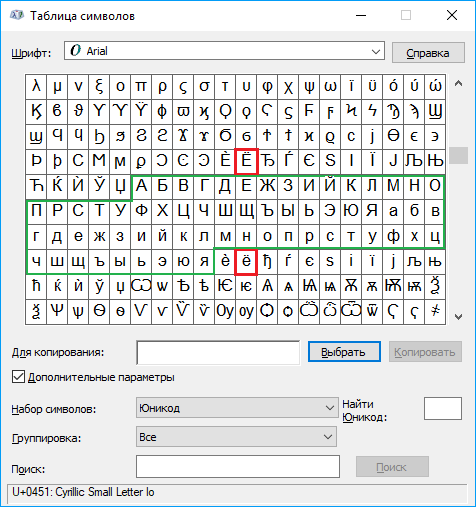
Этот код должен корректно работать для всех вышеприведенных тестов. Тем не менее, в нем еще есть ошибки…

Например, рассмотрим следующий пример:

проёжиков

Наша программа выдаст 2, хотя слово в данном случае, очевидно, одно. В чем проблема?

Если мы запустим программу, отображающую карту символов (charmap), мы увидим, что буквы ё и Ё находятся в стороне от основного поля кириллицы:



Соответственно, необходимо букву ё учитывать отдельно. То же самое относится, например, к украинской букве ї.

Получается, что нам необходимо учитывать достаточно большое количество букв-исключений. А если нам необходимо работать также с арабской письменностью, индусской, корейской, и т.д.? К счастью, существует специальная функция, которая позволяет проверить символ на его принадлежность к одному из символьных диапазонов Unicode. Воспользуемся этой функцией:

String s1 = textBox1.Text;

int count = 0;

int wc = 0;

for (int i = 0; i < s1.Length; i++)

{

if (char.IsLetter(s1[i]))

wc++;

else if (wc > 0)

{

wc = 0;

count++;

}

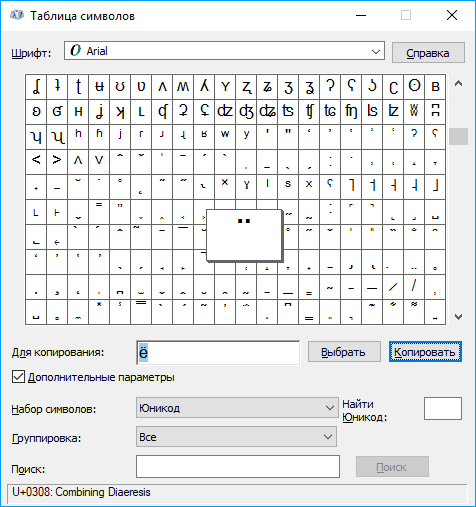
}

if (wc > 0)

count++;

MessageBox.Show(count.ToString());

Данная программа уже работает (почти) полностью корректно. Почему почти? Потому что если мы воспользуемся вариантом представления буквы ё как описано в самом первом примере, программа укажет 2 слова вместо одного. Это можно сделать сформировав символ ё из кириллической буквы е и соответствующей диактрики в программе charmap:



Именно поэтому, в общем случае, нельзя при работе со строками считать, что одно знакоместо соответствует одному символу.

По приведенному примеру самостоятельно составьте приложение, выводящее все целые числа в строке (число — одна или более подряд идущих цифр). Воспользуйтесь для этого функцией char.IsDigit (функция char.IsNumeric также считает и знаки математических операций, дроби, подстрочные/надстрочные индексы и пр.)