Лабораторная работа №6

Тема: Тип данных — строка

Задание: Обработать введенную пользователем строку в соответствии с индивидуальным заданием. Каждое задание обработки должно осуществляться при нажатии соответствующей кнопки. Вывод должен производиться с возможностью копирования и без потери введенной пользователем строки.

Строка — это просто некоторая конечная последовательность символов. Вполне правомерно рассматривать строку как массив символов, но этот подход не учитывает некоторые особенности строк. Прежде всего, значения строк, в отличие от значений массивов, часто используются повторно без изменений. Во-вторых, возможный диапазон значений элемента строки (символа), как правило, невелик. В-третьих, необходимость перестановки отдельных элементов (символов) в строке возникает исключительно редко — если строку отсортировать, скажем, по возрастанию кода символ, весь ее смысл скорее всего потеряется. При сортировке массивов смысл обычно не теряется (там не важны положения элементов относительно друг друга). В-четвертых, для строк вполне определена операция сложения, называемая в данном случае конкатенация.

По этим причинам во многих языках программирования строки выносятся в отдельный тип данных со специальной обработкой. Чаще всего к символам в строке можно обращаться как к элементам массива, например

{

...

String s = "Вося";

Char c = s[1];

...

}

При этом важно помнить, что индексация строк в языке C# всегда начинается с 0, так же, как и у массивов и других составных структур.

Также нужно помнить, что в C# все строки по-умолчанию представлены в кодировке Unicode UTF-16, в которой на один символ может приходиться как один элемент, так и несколько. Символы кириллицы в UTF-16 занимают, как правило, одно знакоместо, но это применимо не ко всем национальным письменностям. Кроме того, для записи символов с диактрическими символами могут использоваться последовательности UTF, например, символ ‘ё’ может быть представлен как буква ‘ё’, так и как буква ‘е’ с модификатором — двумя точкми сверху. Впрочем, полноценный разбор такого рода комбинации выходит за рамки простой работы со строками и требует специализированных библиотек.

Для строк определены следующие основные операции:

1. Операция объединения (конкатенация) строк. Используется стандартная языковая операция сложения:

String S1 = "мама", S2 = "мыла";

S1 = S1 + " " + S2 + " " + "раму";

1. Операция чтения элемента

String s = "Вося";

Char c = s[1];

1. Операции сравнения == !=

Обратите внимание, что стандартный тип строки String не допускает манипуляций с отдельными элементами строк. Для этого существует специальный тип изменяемой строки StringBuilder.

StringBuilder str = new StringBuilder("Вося");

str[1] = 'а';

В типе StringBuilder определены следующие методы:

1. Очистка буфера строки, т.е. стирание всех символов и установка длины в 0:

StringBuilder str = new StringBuilder("Вося");

str.Clear();

1. Вставка текстового представления элемента в строке:

StringBuilder str = new StringBuilder("раз, да еще раз");

str.Insert(12, 100500);

1. Удаление фрагмента заданной длины:

StringBuilder str = new StringBuilder("казнить, нельзя, помиловать");

str.Remove(15, 1);

1. Замена символа (фрагмента) на другой:

StringBuilder str = new StringBuilder("жывотное жывет жывотной жызнью");

str.Replace('ы', 'и');

Во многих случаях необходимо по тому или иному признаку выделить подстроку из строки. Для этого необходимо либо запомнить начало подстроки при разборе строки:

int bi = 0;

String S1 = "Если разобраться, в каждом подобном случае...";

for (int i = 0; i < S1.Length; i++)

if (S1[i] == ',')

bi = i;

String S2 = S1.Substring(0, bi);

В случае StringBuilder:

int bi = 0;

StringBuilder SB1 = new StringBuilder("Если разобраться, в каждом подобном случае...");

for (int i = 0; i < SB1.Length; i++)

if (SB1[i] == ',')

bi = i;

String S2 = SB1.ToString(0, bi);

Либо одновременно копировать символы в некоторую буферную переменную:

String S1 = "Если разобраться, в каждом подобном случае...";

String S2 = "";

for (int i = 0; i < S1.Length; i++)

if (S1[i] == ',')

break;

else

S2 += S1[i].ToString();

Оба этих подхода имеют преимущества и недостатки, первый быстрее работает, второй проще программировать.

Вариант №1

1. Подсчитать количество символов, не являющихся буквами
2. Заменить все запятые на точку с запятой
3. Подсчитать сумму всех чисел в строке
4. Заменить каждую четную букву ‘b’ на ‘б’

Вариант №2

1. Подсчитать количество русских букв
2. Заменить все последовательности ‘abc’ на ‘ПРЕВЕД’
3. Подсчитать количество чисел в строке
4. Заменить каждую 3-ю точку на двоеточие

Вариант №3

1. Подсчитать количество латинских букв
2. Заменить все пробелы на символы подчеркивания
3. Подсчитать среднее арифметическое чисел в строке
4. Заменить каждый второй разделитель на пробел

Вариант №4

1. Подсчитать количество цифр
2. Заменить все последовательности \*\_\* на ^\_^
3. Удвоить все числа в строке