Класс StringBuilder

Создание объекта StringBuilder

1) Пустая строка:
```
StringBuilder sb = new StringBuilder();
```

2) Строке можно задать начальную длину — здесь строке sb будет выделена память под n элементов:
```
int n = 50;
StringBuilder sb = new StringBuilder(n);
```

3) Строка может получить начальное значение:
```
StringBuilder sb = new StringBuilder("qwerty");
```

Консольный ввод-вывод

Для ввода-вывода строки можно использовать методы ReadLine() и WriteLine() класса Console:
```
StringBuilder sb = new StringBuilder(Console.ReadLine());
Console.WriteLine(sb);
```

Обращение к отдельным элементам строки StringBuilder

При работе со строкой типа StringBuilder можно непосредственно обращаться к отдельным элементам этой строки, можно их изменять, при этом память для строки заново не выделяется. Индексы в строке начинаются с 0, т.е. как в обычном массиве. Текущая длина строки хранится в свойстве Length.

***Пример*.** Инвертировать строку, заданную с клавиатуры.
```
StringBuilder sb = new StringBuilder(Console.ReadLine());
char c;
for(int i = 0, j = sb.Length - 1; i < j; i++, j--) {
   c = sb[i];
   sb[i] = sb[j];
   sb[j] = c;
}
Console.WriteLine(sb);
```

Свойства и методы класса StringBuilder

Среди свойств необходимо отметить два наиболее важных:

- Length — текущее количество символов, хранящихся в строке;
- Capacity — максимальное количество символов, которое может храниться в объекте , т.е. для текущего объекта выделена память именно под это количество символов.

Теперь рассмотрим наиболее употребительные методы класса StringBuilder. Обратите внимание на тот факт, что при вызове методов, например: Append(), Insert(), Remove(), Replace() для объекта типа StringBuilder полученный результат сохраняется в текущем объекте. Память для этого объекта будет выделяться заново только в том случае, если текущее количество символов (Length) превысит максимальное (Capacity).

1) **Добавление строки к объекту**

а) добавление строки:

```
StringBuilder sb = new StringBuilder("qwerty");
sb.Append("++");
Console.WriteLine(sb);
```

На экране будет выведено: qwerty++

б) добавление числа (оно автоматически преобразуется в строку):

```
StringBuilder sb = new StringBuilder("qwerty");
sb.Append(12.5);
Console.WriteLine(sb);
```

Результатом будет следующая строка: qwerty12,5

2) **Вставка**

```
StringBuilder sb = new StringBuilder("qwerty");
sb.Insert(3,"357",2);
```

Результат такой операции: qwe357357rty

3) **Удаление**:

```
StringBuilder sb = new StringBuilder("qwerty");
sb.Remove(1,2);
```

В результате получим: qrty

4) **Замена**:

```
StringBuilder sb = new StringBuilder("qwerty_qwerty");
sb.Replace("we","ABC");
```

В результате замены имеем строку: qABCrty_qABCrty

Также можно делать замену отдельных символов, например:

```
StringBuilder sb = new StringBuilder("qwerty_qwerty");
sb.Replace('w','A');
```

В этом случае результат будет таким: qAerty_qAerty

5) **Преобразование** объекта типа `StringBuilder` в строку класса `String`:

```
string s = sb.ToString();
```