Курс “Языки программирования”

Лекция 06.
Работа со строками.
Строки класса String и StringBuilder.

Что мы можем использовать для работы со строками (текстом)?

Два типа строк: String and StringBuilder

System.String

System.String – **класс, специально спроектированный для хранения и обработки строк, предоставляющий возможность выполнять большое количество операций над строками.**

Каждый объект типа String – это неизменяемая последовательность Unicode-символов

```
1.  string greetingText = “Hello from all the guys at GrSU “;

2.  greetingText += “We do hope you enjoy this lesson ”;
```

1. Создается объект типа System.String и инициализируется. .NET runtime выделяет только необходимое количество памяти для хранения этого текста (32 chars)

2. Создается новый объект типа String, выделяется достаточное количество памяти для хранения комбинации двух строк (55 chars). Ссылка на первоначальную строку потеряна.

Объявление и инициализация строк

```
// Простое объявление
string MyString = "Hello World";

// Strings can include escape characters, such as \n or \t, which //
begin with a backslash character (\)
string Path2 = "c:\\Program Files";

// Using verbatim string literals, which start with the at (@)
// symbol.
string Path1 = @"c:\Program Files";

// Call the ToString( ) method on an object and assign the result to
// a string
int myInteger = 235;
string integerString = myInteger.ToString();

// Пустая строка
string myString=String.Empty;
```

Манипуляции со строками

Класс String предоставляет набор методов для манипуляций со строками

Метод или поле	**Назначение**
Chars	Масссив символов строки
Compare( )	Сравнение строк
CpmpareTo()	Сравнение одной строки с другой
Concat()	Конкатенация (объединение) нескольких строк
Copy()	Копирование строки
Join()	Объединение строк из массива строк
Split()	Разбиение строки на подстроки
ToUpper()	Перевод символов строки в верхний регистр
Length	Количество символов в строке
Substring()	Выделение подстроки

Манипуляции со строками

```
int result;
//сравнение двух строк, case sensitive
result = string.Compare(s1, s2);

// сравнение двух строк, ignore case
result = string.Compare(s1, s2, true);

// вставка подстроки в строку
string s10 = s3.Insert(101, "excellent ");

// проверка того, заканчивается ли строка на некоторую
последовательност символов
bool flag = s3.EndsWith("Training"));

// нахождение индекса подстроки
int i = s3.IndexOf("Training");
```

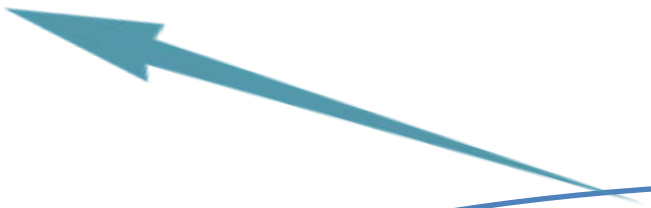
Разбиение строк

```
// некоторая строка
string s1 = "One,Two,Three Liberty Associates, Inc.";

// пробел и запятая, как символы разделители
const char Space = ' ';
const char Comma = ',';

// массив разделителей
char[] delimiters = new char[] { Space, Comma };

// разделяем строки и циклом проходим по массиву найденных подстрок
foreach (string subString in s1.Split(delimiters))
  {
     Console.WriteLine(subString);
  }
```



Результат – массив строк

String - неизменяемые строки

```
string returnNumber = "";
for (int i = 0; i < 1000; i++)
  returnNumber = returnNumber + i.ToString();
```

```
returnNumber = “0”
returnNumber = “1"
returnNumber = “12"
returnNumber = “123"
returnNumber = “1234”
returnNumber = “12345”
returnNumber = “123456”
 . . .
returnNumber = “123456789...999”
```

Создаем строку и в цикле добавляем к ней новые подстроки.

Что происходит?

Фактически создается 999 новых строк !!!

Dynamic Strings (class StringBuilder)

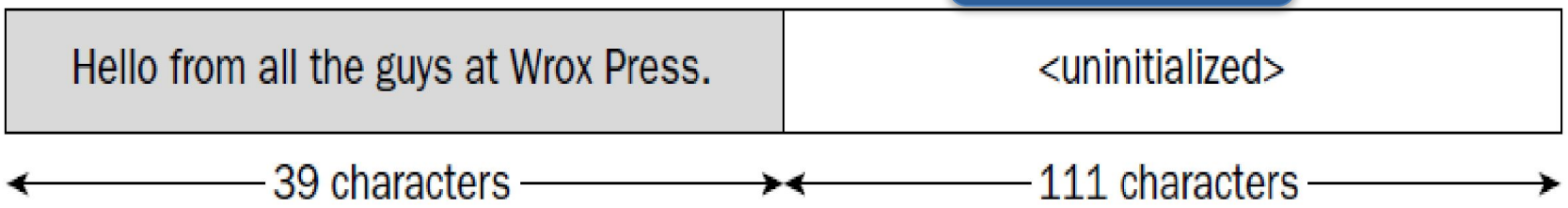
System.Text.StringBuilder class используется для создания изменяемых строк.

StringBuilder это изменяемые строки.

StringBuilder

```
StringBuilder greetingBuilder = new StringBuilder("Hello from all
the guys at Wrox Press. ", 150);
```

Вместимость




```
greetingBuilder.Append("We do hope you enjoy this book as much as
we enjoyed writing it");
```

При вызове метода Append() , добавляемый текст размещается в свободном пространстве, без необходимости выделять большее количество памяти.

В практике, StringBuilder используется для выполнения манипуляций со строками, String - для хранения и отображения результата обработки строки.

StringBuilder

Два основных свойства:
Length – действительная длина строки (количество символов);
Capacity – максимально возможная вместимость строки.

```
StringBuilder sb = new StringBuilder(“Hello”); //Length = 5

StringBuilder sb = new StringBuilder(20); //Capacity = 20

//Capacity можно изменить в любой момент
StringBuilder sb = new StringBuilder(“Hello”);
sb.Capacity = 100;
```

StringBuilder

The following table lists the main StringBuilder methods.

Method	**Purpose**
Append()	Добавление строки
AppendFormat()	Добавление форматированной строки
Insert()	Вставка подстроки
Remove()	Удаление символов из строки
Replace()	Замена подстроки другой подстрокой
ToString()	Приведение к ттипу String

Спасибо за внимание