В.В. Подбельский

Иллюстрации к курсу лекций по дисциплине «Программирование на C#»

07 Часть 2

BigInteger.
DateTime.
StringBuilder

Благодарность Д.Д. Сагалову за помощь в переработке презентаций

Структура BigInteger

BigInteger – неизменяемый тип значений из пространства имён ***System.Numerics***, представляет произвольно большое целое число, значение которого не имеет чёткой верхней или нижней границ. Используется для длинной арифметики, когда встроенных типов недостаточно.

Не имеет границ (MinValue и MaxValue), однако при получении слишком больших объектов возникает ***OutOfMemoryException***.

```
string positiveString = "91389681247993671255432112000000";
BigInteger posBigInt = 0;
try
{
    posBigInt = BigInteger.Parse(positiveString);
    Console.WriteLine(posBigInt); //   9.13896812479936712554321112E+31
}
catch (FormatException)
{
    Console.WriteLine($"Can’t convert {positiveString}");
}
```

https://docs.microsoft.com/en-us/dotnet/api/system.numerics.biginteger?view=net-5.0

Структура DateTime

DateTime – тип значений для представления даты и времени. Используется для представления функционала, связанного со временем в BCL.

Значения времени измеряются в 100-наносекундных единицах, именуемых тактами. Конкретная дата – число тактов с 00:00 1 января 0001 г. по до 23:59:59 31 декабря 9999 г.

Некоторые констркуторы:

```
public DateTime(long ticks);
public DateTime(int year, int month, int day);
public DateTime(int year, int month, int day
                int hour, int minute, int second);
```

Статические поля только для чтения:

MinValue	Минимальное значение типа DateTime.
MaxValue	Максимальное значение типа DateTime.

https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-5.0

Класс StringBuilder

StringBuilder – специальный класс пространства имён ***System.Text***, который представлен как **изменяемая** строку символов. От данного класса нельзя наследоваться.

Используется в сценариях, когда требуется многократное динамическое изменение строк, т. к. расширение буфера StringBuilder не приводит к полному пересозданию строки, что значительно более эффективно, чем использование неизменяемых экземпляров string.

Некоторые конструкторы:

StringBuilder()	Создаёт экземпляр StringBuilder с ёмкостью по умолчанию (16 символов).
StringBuilder(int)	Создаёт экземпляр StringBuilder с указанной ёмкостью.
StringBuilder(string)	Создаёт экземпляр StringBuilder, содержащий заданную строку.

https://docs.microsoft.com/en-us/dotnet/api/system.text.stringbuilder?view=net-5.0

Особенности StringBuilder

Класс **StringBuilder** предоставляет свойства Length и Capacity**:**

- **Length** – реально хранящееся количество символов на данный момент;
- **Capacity** – текущая максимальная ёмкость буфера.

Метод **EnsureCapacity()** позволяет увеличивать текущую ёмкость объекта StringBuilder. При выходе строки за пределы емкости StringBuilder автоматически вызывается метод EnsureCapacity.

Обратите внимание: при установке свойству Length значения меньше длины текущей строки происходит усечение содержимого StringBuilder.

Методы Append и AppendFormat StringBuilder

Класс StringBuilder предоставляет 24 перегруженных метода **Append**, которые дописывают значения разных типов в конец буфера, содержащегося в StringBuilder.

- Предоставляются перегрузки для простых типов, символьных массивов, строк и объектов.

```
public System.Text.StringBuilder Append(int value);
public System.Text.StringBuilder Append(string value);
```

Метод **AppendFormat** предназначен для формирования строки по указанному шаблону и добавления ее в конец строки, содержащейся в StringBuilder.

```
public System.Text.StringBuilder AppendFormat(string value,
                                               params object[] args);
```

Другие Методы StringBuilder

Метод **Insert()** вставляет переданную строку *value* перед символом, позиция которого задана *index*:

```
public System.Text.StringBuilder Insert(int index, string value);
```

Метод **Remove()** удаляет первые *length* элементов, начиная с позиции *startIndex*:

```
public System.Text.StringBuilder Remove(int startIndex, int length);
```

Метод **Replace()** заменяет все вхождения подстроки *old* подстрокой *new*:

```
public System.Text.StringBuilder Replace(string old, string new);
```