Работа с датами и промежутками времени

Структура DateTime

Возможные значения и варианты создания

```
static readonly DateTime MinValue; // 1 янв 1 г.н.э.
static readonly DateTime MaxValue; // 1 янв 10000 г.

DateTime(long ticks);
DateTime(int year, int month, int day[, int hour, int minute, int second[,
    int millisecond]]);
```

Один *такт* (англ. tick) равен 100 наносекундам, или 10^{-7} с.

```
static DateTime Now { get; } // текущая полная дата
static DateTime Today { get; } // только дата

static DateTime Parse(string s[, IFormatProvider p]);
static bool TryParse(string s, out DateTime result);
```

В качестве разделителей для даты можно указывать как точку «.», так и символ «/» (в качестве разделителей для времени необходимо использовать двоеточие «:»), при указании времени можно опускать секунды, при указании *только* даты можно опускать год, между элементами даты и/или времени могут содержаться пробелы.

Свойства

Year, Month, Day, Hour, Minute, Second, Millisecond (все имеют тип int);

`Ticks` (типа `long`) – число тактов (для полуночи 1 января 2001 г. данное свойство равно `631 139 040 000 000 000`);

`Date` (типа `DateTime`) – дата без времени (полночь);

`TimeOfDay` (типа `TimeSpan`) – время без даты;

`DayOfYear` (типа `int`) – число дней от начала года до указанной даты (для 1 февраля будет возвращено `32`);

`DayOfWeek` (типа перечисления `DayOfWeek`) – день недели, соответствующий указанной дате.

Сравнение дат

Можно использовать любые операции сравнения.

Операции над датами

```
DateTime Add(TimeSpan value);
DateTime Subtract(TimeSpan value);
TimeSpan Subtract(DateTime value);
```

Можно использовать операции «+» и «–».

```
DateTime AddYears(int value);
DateTime AddMonths(int value);
DateTime AddDays(double value);
DateTime AddHours(double value);
DateTime AddMinutes(double value);
DateTime AddSeconds(double value);
DateTime AddMilliseconds(double value);
DateTime AddTicks(long value);
```

Параметр `value` может быть как положительным, так и отрицательным.

Форматирование дат

```
string ToString([string fmt][, IFormatProvider p]);
```

ToShortDateString – формат d ("27.01.1756");
ToLongDateString – формат D ("27 января 1756 г.");
ToShortTimeString – формат t ("10:55");
ToLongTimeString – формат T ("10:55:15").

Вспомогательные методы

```
static bool IsLeapYear(int year);
static int DaysInMonth(int year, int month);
```

Структура TimeSpan

Возможные значения, варианты создания и строковое представление

```
static readonly TimeSpan MinValue;
static readonly TimeSpan MaxValue;
    // 10675199.02:48:05 (день.час:мин:сек)
static readonly TimeSpan Zero;

TimeSpan(long ticks);
TimeSpan(int days, int hours, int minutes, int seconds, int milliseconds);
TimeSpan([int days,] int hours, int minutes, int seconds);
static TimeSpan FromDays(double value);
static TimeSpan FromHours(double value);
static TimeSpan FromMinutes(double value);
static TimeSpan FromSeconds(double value);
static TimeSpan FromMilliseconds(double value);
static TimeSpan FromTicks(long value);

static TimeSpan Parse(string s);
static bool TryParse(string s, out TimeSpan result);
```

Строка `s` должна иметь следующий формат (необязательные элементы заключены в квадратные скобки): `[-][d.]h:m[:s[.f]]`, Допускается также строка, содержащая только количество дней (возможно, со знаком «–»). Строка `s` может также содержать начальные и конечные пробелы.

Приведенный выше формат используется методом `ToString` для обратного преобразования объекта типа `TimeSpan` к строковому представлению. В полученной строке обязательно присутствуют элементы `h`, `m` и `s`, причем для каждого из них отводится по две цифры. Элементы `d` и `f` указываются только в случае, если они не равны 0, а знак «–» — если промежуток времени является отрицательным.

Свойства

Days, Hours, Minutes, Seconds, Milliseconds (все имеют тип int);
TotalDays, TotalHours, TotalMinutes, TotalSeconds, TotalMilliseconds (все имеют тип double);
Ticks типа long.

Сравнение промежутков времени

Можно использовать любые операции сравнения.

Операции над промежутками времени

```
TimeSpan Add(TimeSpan value);
```

```
TimeSpan Subtract(TimeSpan value);
TimeSpan Negate(); // меняет знак на противоположный
TimeSpan Duration(); // отбрасывает знак
```

Вместо этих методов (кроме Duration) можно использовать операции «+» и «–», в том числе унарный минус.