**Форматирование даты и времени**

Помимо числовых значений, форматированию нередко подлежит и другой тип

данных: DateTime. Это структура, представляющая дату и время. Значения даты и

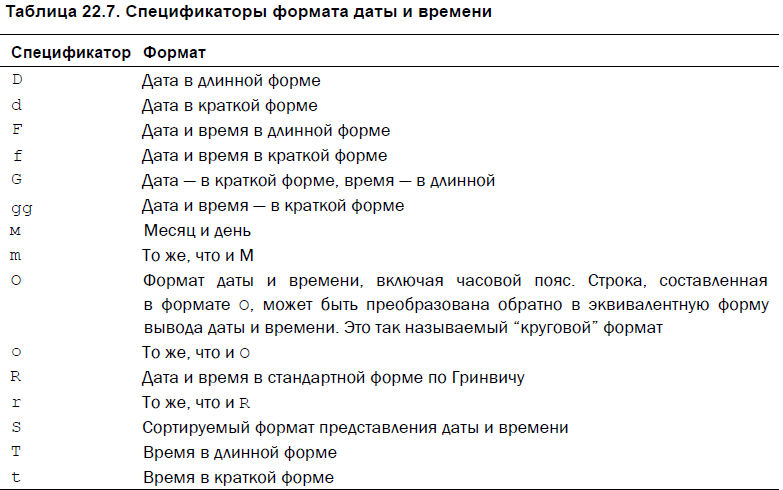
времени могут отображаться самыми разными способами.

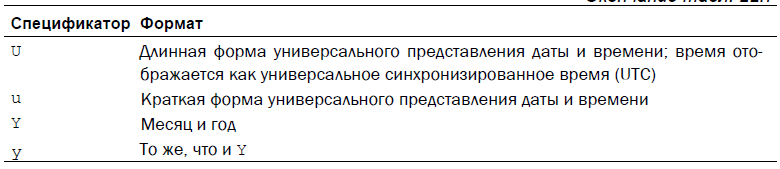
Форматирование даты и времени осуществляется с помощью спецификаторов

формата. Спецификаторы формата даты и времени сведены в табл. 22.7. Конкретное

представление даты и времени может отличаться в силу региональных и языковых особенностей

и поэтому зависит от настройки параметров культурной среды.





В приведенном ниже примере программы демонстрируется применение спецификаторов

формата даты и времени.

(***glava22\_14***)

class TimeAndDateFormatDemo

{

static void Main()

{

DateTime dt = DateTime.Now;

Console.WriteLine("Format d: {0:d}", dt);

Console.WriteLine("Format D: {0:D}", dt);

Console.WriteLine("Format t: {0:t}", dt);

Console.WriteLine("Format T: {0:T}", dt);

Console.WriteLine("Format f: {0:f}", dt);

Console.WriteLine("Format F: {0:F}", dt);

Console.WriteLine("Format g: {0:g}", dt);

Console.WriteLine("Format G: {0:G}", dt);

Console.WriteLine("Format m: {0:m}", dt);

Console.WriteLine("Format M: {0:M}", dt);

Console.WriteLine("Format o: {0:o}", dt);

Console.WriteLine("Format O: {0:O}", dt);

Console.WriteLine("Format r: {0:r}", dt);

Console.WriteLine("Format R: {0:R}", dt);

Console.WriteLine("Format s: {0:s}", dt);

Console.WriteLine("Format u: {0:u}", dt);

Console.WriteLine("Format U: {0:U}", dt);

Console.WriteLine("Format y: {0:y}", dt);

Console.WriteLine("Format Y: {0:Y}", dt);

}

}

В следующем примере программы воспроизводятся очень простые часы. Время обновляется

каждую секунду, и каждый час компьютер издает звонок. Для получения отформатированного

строкового представления времени перед его выводом в этой программе

используется метод ToString() из структуры DateTime. Через каждый час

символ звукового предупреждающего сигнала присоединяется к отформатированной

строке, представляющей время, в результате чего звенит звонок.

(***glava22\_15***)

class SimpleClock

{

static void Main()

{

string t;

int seconds;

DateTime dt = DateTime.Now;

seconds = dt.Second;

for(; ; )

{

dt = DateTime.Now;

//every second update

if(seconds != dt.Second)

{

seconds = dt.Second;

t = dt.ToString("T");

if (dt.Minute == 0 && dt.Second == 0)

t = t + "\a";

Console.WriteLine(t);

}

}

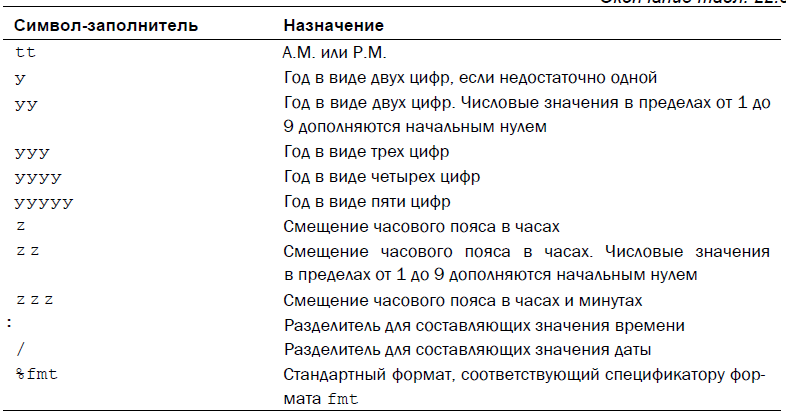
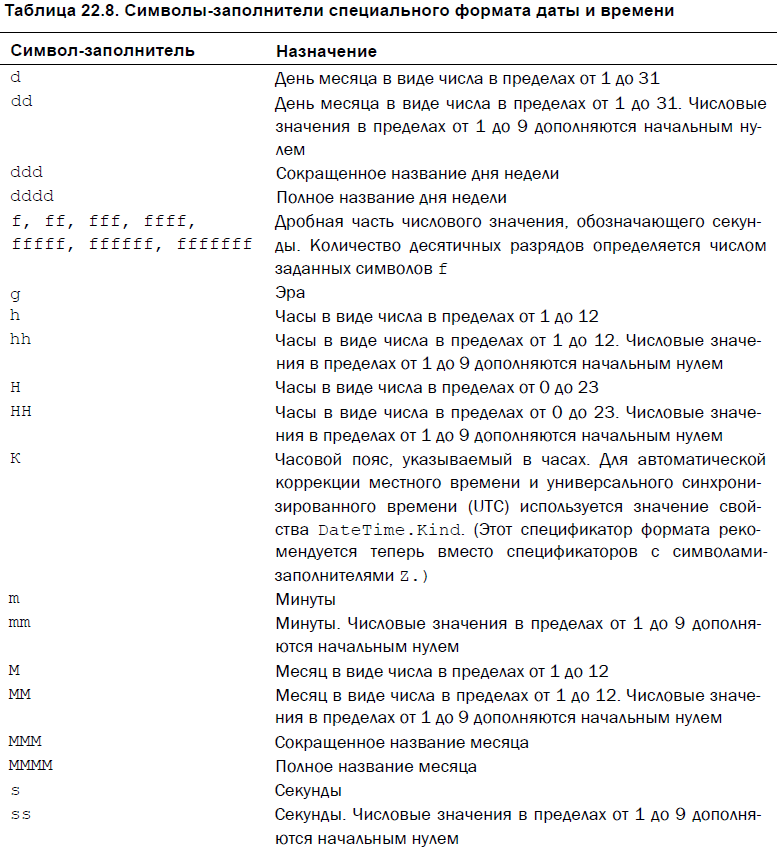
}

}

**Определение пользовательского формата даты и времени**

Для определения пользовательского

формата даты и времени служат символы-заполнители, перечисленные в табл. 22.8.



Вообще говоря, если один из этих символов указывается отдельно, то он интерпретируется

как спецификатор формата. В противном случае он считается символом-

заполнителем. Поэтому если требуется указать несколько таких символов отдельно,

но интерпретировать их как символы-заполнители, то перед каждым из них следует

поставить знак %.

В приведенном ниже примере программы демонстрируется применение нескольких

форматов даты и времени.

(***glava22\_16***)

class CustomTimeAndFateFormatsDemo

{

static void Main()

{

DateTime dt = DateTime.Now;

Console.WriteLine("Time: {0:hh:mm tt}", dt);

Console.WriteLine("Time 24h: {0:HH:mm}", dt);

Console.WriteLine("Date: {0:ddd MMM dd, yy}", dt);

Console.WriteLine("Age: {0:gg}", dt);

Console.WriteLine("Time in seconds: {0:HH:mm:ss tt}", dt);

Console.WriteLine("Day and month format m: {0:m}", dt);

Console.WriteLine("Minutes format m: {0:%m}", dt);

}

}

**Форматирование промежутков времени**

Начиная с версии 4.0, в среде .NET Framework появилась возможность форматировать

объекты типа TimeSpan — структуры, представляющей промежуток времени.

Объект типа TimeSpan может быть получен самыми разными способами, в том числе

и в результате вычитания одного объекта типа DateTime из другого. И хотя форматировать

объекты типа TimeSpan приходится нечасто, о такой возможности все же стоит

упомянуть вкратце.

По умолчанию в структуре TimeSpan поддерживаются три стандартных спецификатора

формата даты и времени: с, g и G. Они обозначают инвариантную форму

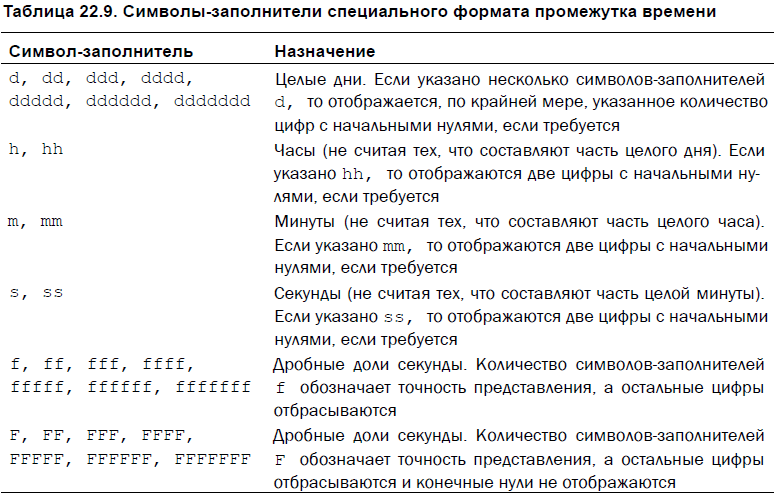
промежутка времени, короткую и длинную форму с учетом культурной среды соответственно

(последняя форма всегда включает в себя дни). Кроме того, в структуре

TimeSpan поддерживаются специальные спецификаторы формата даты и времени,

приведенные в табл. 22.9. Вообще говоря, если один из этих спецификаторов используется

в отдельности, его нужно предварить символом %.



В приведенной ниже программе демонстрируется форматирование объектов типа

TimeSpan на примере отображения времени, которое приблизительно требуется для

вывода на экран 1000 целых значений в цикле for.

(***glava22\_17***)

class TimeSpanDemo

{

static void Main()

{

DateTime start = DateTime.Now;

//numbers 1 - 1000

for(int i = 1; i <= 1000; i++)

{

Console.Write(i + " ");

if ((i % 10) == 0) Console.WriteLine();

}

Console.WriteLine();

DateTime end = DateTime.Now;

TimeSpan span = end - start;

Console.WriteLine("Maintain time: {0:c}", span);

Console.WriteLine("Maintain time: {0:g}", span);

Console.WriteLine("Maintain time: {0:G}", span);

Console.WriteLine("Maintain time: 0.{0:fff} seconds", span);

}}

Выполнение этой программы приводит к следующему результату, который и в

этом случае зависит от конкретных настроек языковых и региональных параметров

локализации базового программного обеспечения, а также от загрузки системы задачами

и ее быстродействия.

Maintain time: 00:00:00.0516401

Maintain time: 0:00:00,0516401

Maintain time: 0:00:00:00,0516401

Maintain time: 0.051 seconds