Manual del Tiempo Continuo

Daniel Rus

22 de septiembre de 2005

Parte I

Referencia de las APIs

Capítulo 1

API de CTime

1.1. Constantes

Now El instante actual en segundos

TheBegin El comienzo de los tiempos

TheEnd El final de los tiempos

Today El instante actual en dias

UnknownCTime El momento desconocido

1.2. Funciones Built-In

EasterIn(CTime tmi)

Devuelve el día que representa la Pascua en un año dado. Cualquier `CTime` puede ser usado como argumento.

EasterInYear(Real año)

Devuelve el día que representa la Pascua en un año dado.

NewCTime(Real año
[, Real mes=-1, Real dia=-1, Real hora=-1, Real minuto=-1, Real segundo=-1])

Devuelve un nuevo `CTime` basado en los argumentos dados. La base temporal del `CTime` creado será establecida con el último parámetro válido de la lista de argumentos.
Ejemplos:

```
CTime tmi1 = NewTime(2005,1,2);  -> y2005m1d2 (Daily)
CTime tmi2 = NewTime(2005,0);    -> y2005 (Yearly)
CTime tmi3 = NewTime(2005,3,33); -> Error
```

Add(CTime tmi, Real año
[, Real mes, Real dia, Real hora, Real minuto, Real segundo])

Devuelve un `CTime` como resultado de añadir los parámetros `Real` dados al parámetro `CTime tmi`. Solo los parámetros relacionados con la granularidad del parámetro `tmi` serán aplicados:

`Real año` ->si tmi tiene base Anual
`Real año y mes` ->si tmi tiene base Mensual
`Real año, mes y dia` ->si tmi tiene base Diaria
`Real año, mes, dia y hora` ->si tmi tiene base Horaria
`Real año, mes, dia, hora y minuto` ->si tmi tiene base Minutal
`Real año, mes, dia, hora, minuto y segundo` ->si tmi tiene base Secundal

`Biggest((CTime tmi1, CTime tmi2, [CTime tmi3, ...])`

Devuelve el instante de tiempo más grande de la lista. Cuando dos o más parámetros tienen el mismo tamaño, devuelve el primero de ellos.
Ejemplo:
```
CTime x = Biggest(y1996m1d1, y1986m10d5h22, y1952, y1973m9d5);
```
Resultado: `x == y1952`

`Smallest(CTime tmi1, CTime tmi2 [, CTime tmi3, ...])`

Devuelve el instante de tiempo más pequeño de la lista. Cuando dos o más parámetros tienen el mismo tamaño, devuelve el primero de ellos.
Ejemplo:
```
CTime x = Smallest(y1996m1d1, y1986m10d5h22, y1952, y1973m9d5);
```
Resultado: `x == y1986m10d5h22`

`Sooner(CTime tmi1, CTime tmi2 [, CTime tmi3, ...])`

Devuelve el instante de tiempo de la lista que ocurre antes.
Ejemplo:
```
CTime x = Sooner(y1996m1d1, y1986m10, y1952, y1973m9d5);
```
Resultado: `x == y1952`

`Later(CTime tmi1, CTime tmi2, [CTime tmi3, ...])`

Devuelve el instante de tiempo de la lista que ocurre más tarde.
Ejemplo:
```
CTime x = First(y1996m1d1, y1986m10, y1952, y1973m9d5);
```
Resultado: `x == y1952`

`ContainerOf(CTime tmi)`

Devuelve el CTime que contiene al CTime dado.

`Maximal(CTime tmi, CTimeSet dating)`

Devuelve el instante de tiempo maximal del CTime dado como parámetro, en el conjunto temporal dating.
Ejemplo:
```
CTime x = Maximal(y1996m1d3, cMonth(1));
```
Resultado: `x == y1996m1`

MaximalSucc(CTime tmi, CTimeSet dating, [Real n=1])

Devuelve el enésimo instante de tiempo maximal posterior al dado como parámetro, en el conjunto temporal dating. Si n es negativo devolverá el enésimo predecesor maximal.

Ejemplo:

```
CTime x = MaximalSucc(y1996m1d1, CTSAllMonths, 2);
```

Resultado: `x == y1996m3`

MaximalPred(CTime tmi, CTimeSet dating, [Real n=1])

Devuelve el enésimo instante de tiempo maximal anterior al dado como parámetro, en el conjunto temporal `dating`. Si `n` es negativo devolverá el enésimo sucesor maximal.

Ejemplo:

```
CTime x = MaximalPred(y1996m1d1, CTSAllMonths, 2);
```

Resultado: `x == y1996m1`

Succ(CTime tmi, CTimeSet dating, [Real n=1])

Devuelve el enésimo instante de tiempo posterior al dado como parámetro, en el conjunto temporal dating. Si `n` fuera negativo devolverá el enésimo predecesor.

Ejemplo:

```
CTime x = Succ(y1996m1d1, Day(1), 5);
```

Resultado: `x == y1996m6d1`

Pred(CTime tmi, CTimeSet dating [, Real n=1])

Devuelve el enésimo instante de tiempo anterior al dado como parámetro, en el conjunto temporal dating. Si `n` fuera negativo devolverá el enésimo sucesor.

Ejemplo:

```
CTime x = Pred(y1996m1d6, Day(1), 5);
```

Resultado: `x == y1996m6d1`

NextCTime(CTime tmi)

Devuelve el instante de tiempo sucesor del instante de tiempo dado.

Ejemplo:

```
CTime x = Succ(y2004m10);
```

Resultado: `x == y2004m11`

PrevCTime(CTime tmi)

Devuelve el instante de tiempo sucesor del instante de tiempo dado.

Ejemplo:

```
CTime x = PrevCTime(y2004m10);
```

Resultado: `x == y2004m9`

First(CSeries cser)

Devuelve el instante CTime inicial de una serie.

Last(CSeries cser)

Devuelve el instante CTime final de una serie.

CTimeFromText(Text tmi, Text format)

La función `CTimeFromText` es la función inversa de `TextFromCTime` y convierte la cadena de caracteres del parámetro `Text tmi` en un valor `CTime`, utilizando el formato especificado por el parámetro `Text format`, que es una cadena de caracteres consistente en descriptores de campos y caracteres de texto.

Cada descriptor de campo consiste en un carácter de porcentaje `%` seguido por otro carácter que especifica el reemplazo para el descriptor de campo. Todos los otros caracteres de la cadena `format` deben tener un carácter concordante en la cadena de entrada, salvo los espacios en blanco que pueden concordar con cero o más espacios en blanco de la cadena de entrada. Entre dos descriptores de campo cualesquiera debe haber siempre un espacio en blanco o bien otros caracteres alfanuméricos.

La función procesa la cadena de entrada de izquierda a derecha. Cada uno de los tres posibles elementos de entrada (espacio en blanco, literal o formato) se tratan uno detrás de otro. Si no se puede hacer coincidir la entrada con la cadena de formato, la función se detiene. El resto de las cadenas de formato y de entrada no se procesa.

Los descriptores de campo de entrada se listan a continuación. En caso de que la concordancia sea con cadenas de texto (como el nombre de un día de la semana o de un mes), la comparación no tiene en cuenta las mayúsculas. En caso de que la concordancia sea con números, se permite poner ceros al principio aunque no es obligatorio.

%% El carácter `%`.

%a o %A El nombre del día de la semana según la localización actual, en forma abreviada o completa.

%b o %B o %h El nombre del mes según la localización actual, en forma abreviada o completa.

%c La representación de fecha y hora para la localización actual.

%C El número de siglo (0-99).

%d o %e El día del mes (1-31).

%D Equivalente a `%m/ %d/ %y`. (Este es el estilo americano para las fechas, muy confuso para los no americanos, especialmente debido a que el formato `%d/ %m/ %y` es el que se usa ampliamente en Europa. El formato del estándar ISO 8601 es `%Y- %m- %d`.)

%H La hora (0-23).

%I La hora en formato de 12 horas (1-12).

%j El número de día del año (1-366).

%m El número de mes (1-12).

%M El minuto (0-59).

%n Cadena arbitraria de espacios en blanco.

%p El equivalente de AM o PM en la localización. (Nota: puede no existir.)

%r La hora en formato de 12 horas (usando el equivalente de AM o PM en la localización). En la localización POSIX equivale a `%I: %M: %S %p`. Si el campo `t_fmt_ampm` está vacío en la categoría `LC_TIME` de la localización actual el comportamiento es indefinido.

%R Equivalente a `%H: %M`.

%S El segundo (0-60; 60 puede darse para segundos de salto (leap seconds); anteriormente también se permitía 61).

%t Cadena arbitraria de espacios en blanco.

%T Equivalente a `%H:%M:%S`

%U El número de semana tomando el domingo como primer día de la semana (0-53). El primer domingo de enero es el primer día de la semana 1.

%w El número de día de la semana (0-6) con domingo = 0.

%W El número de semana tomando el lunes como primer día de la semana (0-53). El primer lunes de enero es el primer día de la semana 1.

%x La fecha, usando el formato de fecha de la localización.

%X La hora, usando el formato de hora de la localización.

%y El año dentro del siglo (0-99). Cuando no se especifica el siglo, los valores comprendidos en el rango 69-99 se refieren al siglo 20 (1969-1999) y los valores comprendidos en el rango 00-68 se refieren al siglo 21 (2000-2068).

%Y El año, incluyendo el siglo (por ejemplo, 1991).

Algunos descriptores de campo pueden ser modificados por los caracteres modificadores `E` u `O` para indicar el uso de un formato o especificación alternativo. Si el formato o especificación alternativo no existe en la localización actual, se utiliza el descriptor de campo sin modificar.

El modificador `E` especifica que la cadena de entrada puede contener versiones alternativas de la representación de fecha y hora dependientes de la localización:

%Ec La representación alternativa de fecha y hora de la localización.

%EC El nombre del año base (período) en la representación alternativa de la localización.

%Ex La representación alternativa de fecha de la localización.

%EX La representación alternativa de hora de la localización.

%Ey El desplazamiento desde %EC (sólo año) en la representación alternativa de la localización.

%EY La representación alternativa completa para el año.

El modificador `O` especifica que la entrada numérica puede estar en un formato alternativo dependiente de la localización:

%Od o %Oe El día del mes usando los símbolos numéricos alternativos de la localización; los ceros del comienzo están permitidos pero no son obligatorios.

%OH La hora (foramto 24 horas) usando los símbolos numéricos alternativos de la localización.

%OI La hora (formato 12 horas) usando los símbolos numéricos alternativos de la localización.

%Om El mes usando los símbolos numéricos alternativos de la localización.

%OM Los minutos usando los símbolos numéricos alternativos de la localización.

%OS Los segundos usando los símbolos numéricos alternativos de la localización.

%OU El número de semana del año (tomando el domingo como primer día de la semana) usando los símbolos numéricos alternativos de la localización.

%Ow El número del día de la semana (domingo=0) usando los símbolos numéricos alternativos de la localización.

%OW El número de semana del año (tomando el lunes como primerdía de la semana) usando los símbolos numéricos alternativos de la localización.

%Oy El año (desplazamiento desde %C) usando los símbolos numéricos alternativos de la localización.

`LastUpdate(Text camino)`

Devuelve el momento de última modificación del fichero dado.

Capítulo 2

API de CTimeSet

2.1. Constantes

CTEmpty The empty CTimeSet. No instants of time

CTInYears The CTimeSet of the whole Time given in a YEARS granularity

CTInMonths The CTimeSet of the whole Time given in a MONTHS granularity

CTInDays The CTimeSet of the whole Time given in a DAYS granularity

CTInHours The CTimeSet of the whole Time given in a HOURS granularity

CTInMinutes The CTimeSet of the whole Time given in a MINUTES granularity

CTInSeconds The CTimeSet of the whole Time given in a SECONDS granularity

CTSEaster El conjunto temporal de todos los domingos de Pascua. Las fechas incluidas entre el 1250 AC. y el 1582 DC. corresponden a la Pascua Judía o Pésaj calculada en el calendario Juliano. A partir del 1583 DC. corresponden con la Pascua del calendario Gregoriano.

2.2. Funciones Built-In

2.2.1. Fechados Básicos

Year(Real year)

Devuelve el `CTimeSet` compuesto por todos los instante de tiempo del año indicado como parámetro.

Month(Real month)

Devuelve el `CTimeSet` compuesto por todos los instante de tiempo del mes indicado como parámetro.

Ejemplos:

```
CTimeSet month2 = Month(2);
```

Day(Real month)

Devuelve el `CTimeSet` compuesto por todos los instante de tiempo del día indicado como parámetro. Si el día indicado es el 31, solo el día 31 de los meses con 31 días estará incluído. Es posible crear un CTimeSet que represente el último día de cada mes usando el valor negativo -1.

Ejemplos:

```
CTimeSet stDay = Day(1);
CTimeSet lastDay = Day(-1);
```

Hour(Real hour)

Devuelve el `CTimeSet` compuesto por todos los instante de tiempo de la hora dada como parámetro.

Ejemplos:

```
CTimeSet h0 = Hour(0);
CTimeSet h12 = Hour(12);
CTimeSet h23 = Hour(23);
```

Minute(Real minute)

Devuelve el `CTimeSet` compuesto por todos los instante de tiempo del minuto dado como parámetro.

Ejemplos:

```
CTimeSet m0 = Minute(0);
CTimeSet m30 = Minute(30);
CTimeSet m59 = Minute(59);
```

Second(Real second)

Devuelve el `CTimeSet` compuesto por el segundo dado.

Ejemplos:

```
CTimeSet s0 = Second(0);
CTimeSet s30 = Second(30);
CTimeSet s59 = Second(59);
```

WD(Real weekday)

Devuelve el `CTimeSet` compuesto por todos los instantes de tiempo que son el enésimo día de la semana. El parámetro `n` tiene que estar entre los valores 1 (Lunes) y 7 (Domingo).

2.2.2. Conjuntos Temporales Finitos

Inside(CTime tmi)

Devuelve el `CTimeSet` formado por el único instante de tiempo dado.

Ejemplos:

```
CTimeSet aY = Inside(y2008);
CTimeSet aM = Inside(y2008m2);
CTimeSet aD = Inside(y2008m2d29);
CTimeSet aH = Inside(y2008m2d29h10);
CTimeSet aMi = Inside(y2008m2d29h10i10);
CTimeSet aS = Inside(y2008m2d29h10i10s35);
```

Interval(CTime from, CTime to)

Devuelve el `CTimeSet` formado por todos los instantes de tiempo entre `from` y `to`. La granularidad básica de todos los instantes de tiempo se toma de la granularidad del segundo parámetro.

Ejemplo:

```
CTimeSet daysOfYear2005 = In(y2005, y2005m12d31);
CTime ini = y2008m02;
CTime end = y2008m03d01h10;
CTimeSet interval = Interval(ini, end);
```

CTimesOfSet(Set setOfCTimes)

Devuelve el `CTimeSet` formado por todos los instantes de tiempo contenidos en el parámetro `setOfCTimes`.

Ejemplo:

```
Set ctimeSet = SetOfCTime(y2000, y2001m7, y2004m9d5, y2005m11, y2005m12d24h21);
CTimeSet ctms = CTimesOfSet(ctimeSet);
```

2.2.3. Operaciones Booleanas

CTimeSet + CTimeSet

Devuelve la unión de ambos conjuntos temporales.

CTimeSet * CTimeSet

Devuelve la intersección de ambos conjuntos temporales.

Ejemplos:

```
CTimeSet cts1 = WD(1) * WD(2);
CTimeSet cts2 = (WD(1) * (Hour(10) + Hour(11))) * (WD(2) * (Hour(20) +
Hour(21)));
CTimeSet cts3 = WD(1) * (WD(2) + Hour(10));
CTimeSet cts4 = Month(1) * Interval(y2000, y2010);
```

CTimeSet - CTimeSet

Devuelve la diferencia entre ambos conjuntos temporales.

Ejemplos:

```
CTimeSet cts1 = Inside(y2006) - Inside(y2006m1);
CTimeSet cts2 = Year(2006) - Month(1);
CTimeSet cts3 = Year(2006) - Hour(10);
```

2.2.4. Operadores de Traslación

Succ(CTimeSet ctmsSrc, Real niter, CTimeSet ctmsDest)

SelfSucc(CTimeSet ctms, Real niter)

▷ SelfSucc sobre un Second:

```
CTimeSet second40 = Second(40);
CTimeSet s0  = SelfSucc(second40, -40);
CTimeSet s30 = SelfSucc(second40, -10);
CTimeSet s59 = SelfSucc(second40,  19);
```

```
resultado:
s0  == segundo  0 de cada minuto
s30 == segundo 30 de cada minuto
s59 == segundo 59 de cada minuto
```

▷ SelfSucc sobre un Minute:

```
CTimeSet minute40 = Minute(40);
CTimeSet m0  = SelfSucc(minute40, -40);
CTimeSet m30 = SelfSucc(minute40, -10);
CTimeSet m59 = SelfSucc(minute40,  19);
resultado:
m0  == minuto  0 de cada hora
m30 == minuto 30 de cada hora
m59 == minute 59 de cada hora
```

▷ SelfSucc sobre un Hour:

```
CTimeSet hour10 = Hour(10);
CTimeSet h0  = SelfSucc(hour10, -10);
CTimeSet h12 = SelfSucc(hour10, 2);
CTimeSet h23 = SelfSucc(hour10, 13);
resultado:
h0  == Hora 0 de cada día
h12 == Hora 12 de cada día
h23 == Hora 23 de cada día
```

▷ SelfSucc sobre un Day:

```
CTimeSet each30th = Day(30);
CTimeSet day1  = SelfSucc(each30th, 2);
CTimeSet day15 = SelfSucc(each30th, -15);
CTimeSet day31 = SelfSucc(each30th, 1);
resultado:
day1 == día 1 de cada mes
day15 == día 15 de cada mes
day31 == día 31 de cada mes
```

▷ SelfSucc sobre un Month:

```
CTimeSet month1 = Month(1);
CTimeSet month2 = SelfSucc(month1, 1);
resultado: febrero
```

▷ SelfSucc sobre un Year:

```
CTimeSet year1975 = Year(1975);
CTimeSet year1973 = SelfSucc(year1975, -2);
resultado: y1973
```

▷ SelfSucc sobre un WD:

```
CTimeSet sn = WD(7);
CTimeSet th = SelfSucc(sn, -3);
resultado: jueves
```

▷ SelfSucc sobre un CTimesOfSet:

```
Set ctimeSet = SetOfCTime(y2000,y2001m2);
CTimeSet ctms = CTimesOfSet(ctimeSet);
CTimeSet selfSucc = SelfSucc(ctms, 10);
resultado: y2001m12 + y2010
```

▷ SelfSucc sobre un Inside:

```
CTimeSet Y   = Inside(y2010);
CTimeSet aY  = SelfSucc(Y, -2);
resultado:   *         *   *      year 2008  **:**:**
CTimeSet M   = Inside(y2008m01);
CTimeSet aM  = SelfSucc(M, 1);
resultado:   *         *   febrero,    2008  **:**:**
CTimeSet D   = Inside(y2008m03d3);
CTimeSet aD  = SelfSucc(D, -3);
resultado: viernes     29. febrero,    2008  **:**:**
CTimeSet H   = Inside(y2008m2d28h23);
CTimeSet aH  = SelfSucc(H, 11);
resultado: viernes     29. febrero,    2008  10h
CTimeSet Mi  = Inside(y2008m2d29h10i20);
CTimeSet aMi = SelfSucc(Mi, -10);
resultado: viernes     29. febrero,    2008  10:10
CTimeSet S   = Inside(y2008m2d29h10i10s55);
CTimeSet aS  = SelfSucc(S, -20);
resultado: viernes     29. febrero,    2008  10:10:35
```

▷ SelfSucc sobre un Interval:

```
CTime ini  = y2008m02;
CTime end  = y2008m03d01h10;
CTimeSet in = Interval(ini, end);
CTimeSet succIn = SelfSucc(in, 24);
resultado: [ y2008m02d02 .. y2008m03d02h10 ]
```

▷ SelfSucc sobre una Unión de Conjuntos:

```
CTimeSet losDomingos = WD(7);
CTimeSet hora9 = Hour(9);
CTimeSet domingos_u_hora9 = losDomingos + hora9;
CTimeSet ctms = SelfSucc(domingos_u_hora9, 1);
resultado: lunes + 10h
```

▷ SelfSucc sobre una Intersección de Conjuntos:

```
CTimeSet cts1 = WD(7) * (WD(1) + Hour(9));
CTimeSet cts2 = Month(2) * Interval(y2001, y2011);
CTimeSet cts3 = SelfSucc(cts1, 1);
resultado: WD(1) * (WD(2) + Hour(10))
CTimeSet cts4 = SelfSucc(cts2, -1);
resultado: Month(1) * In(y2000, y2010)
```

▷ SelfSucc sobre una Diferencia de Conjuntos:

```
CTimeSet prCts1 = Inside(y2005) - Inside(y2005m12);
CTimeSet cts1 = SelfSucc(prCts1, 1);
resultado: Inside(y2006) - Inside(y2006m1);
```

```
CTimeSet prCts2 = Year(2004) - Month(11);
CTimeSet cts2 = SelfSucc(prCts2, 2);
resultado: Year(2006) - Month(1);
CTimeSet prCts3 = Year(2010) - Hour(14);
CTimeSet cts3 = SelfSucc(prCts3, -4);
resultado: Year(2006) - Hour(10);
```

Todas las operaciones anteriores admiten además las operaciones **SelfSucc** y **SelfRange**.

SuccInG(CTimeSet ctms, Real niter, Text granularity)

Desplaza el Conjunto Temporal **ctms, niter** unidades en la granularidad **granularity** dada, que puede tomar los valores "**SECONDS**", "**MINUTES**", "**HOURS**", "**DAYS**", "**MONTHS**", "**YEARS**".

▷ SuccInG sobre un CTimesOfSet:

```
Set ctimeSet = SetOfCTime(y2000,y2001m2);
CTimeSet ctms = CTimesOfSet(ctimeSet);
CTimeSet succInG = SuccInG(ctms, 10, "DAYS");
resultado: [ y2000m01d11 .. y2001m01d10 ] + [ y2001m02d11 .. y2001m03d10 ]
```

▷ SuccInG sobre un CTimesOfSet:

```
CTimeSet aDa  = SuccInG(Day(31),  10, "HOURS");
resultado: desde las 10h de cada día 31 hasta las 9h de cada día 1
CTimeSet aDb  = SuccInG(Day(1), -743, "HOURS");
resultado: desplaza 743 horas hacia atrás cada día 1 de mes
```

SelfRange(CTimeSet ctms, Real initIter, Real endIter)

▷ SelfRange sobre un Second:

```
CTimeSet s25to35 = SelfRange(Second(25), 0, 10);
resultado: [ segundo 25 .. segundo 35 ]
```

▷ SelfRange sobre un Minute:

```
CTimeSet m50to10 = SelfRange(Minute(0), -10, 10);
resultado: [ minuto 50 .. minuto 10 ]
```

▷ SelfRange sobre un Hour:

```
CTimeSet hour0 = Hour(0);
CTimeSet h10to20 = SelfRange(hour0, 10, 20);
resultado: [ hora 10 .. hora 20 ]
```

▷ SelfRange sobre un Day:

```
CTimeSet each1st = Day(1);
CTimeSet day25to5 = SelfRange(each1st, 24, 35);
resultado: del día 25 al 5 de cada mes
```

▷ SelfRange sobre un Month:

```
CTimeSet month1   = Month(1);
CTimeSet month7a8 = SelfRange(month1, 6, 7);
resultado: julio + agosto
```

▷ SelfRange sobre un Year:

```
CTimeSet year19      = Year(1900);
CTimeSet year20b2004 = SelfRange(year19, 100, 104);
resultado: [ y2000 .. y2004 ]
```

▷ SelfRange sobre un WD:

```
CTimeSet wend = SelfRange(WD(6), 0, 1);
resultado: sábado + domingo
```

▷ SelfRange sobre un CTimesOfSet:

```
Set setTimes = [[ CTime y2005m07d30, CTime y2005m07d31 ]];
CTimeSet rCTSet = SelfRange(CTimesOfSet(setTimes), 7, 14);
resultado: [ y2005m08d06 .. y2005m08d14 ]
```

▷ SelfRange sobre un conjunto Inside (I):

```
CTimeSet Y = Inside(y2010);
CTimeSet aY = SelfRange(Y, -10, 0);
resultado: [ y2000 .. y2010 ]
```

▷ SelfRange sobre un conjunto Inside (II):

```
CTimeSet H = Inside(y2008m2d28h23);
CTimeSet aH = SelfRange(H, 11, 12);
resultado: [ y2008m2d29h10 .. y2008m2d29h11 ]
```

▷ SelfRagne sobre un conjunto Interval:

```
CTimeSet rIn = SelfRange(Interval(y2005m07d30, y2005m07d31), 7, 14);
resultado: [ y2005m08d06 .. y2005m08d14 ]
```

▷ SelfRange sobre una unión de conjuntos:

```
CTimeSet losDomingos = WD(7);
CTimeSet hora9 = Hour(9);
CTimeSet domingos_u_hora9 = losDomingos + hora9;
CTimeSet ctms = SelfRange(domingos_u_hora9, 1, 5);
resultado:
lunes + martes + miércoles + jueves + viernes + [ 10h .. 14h ]
```

▷ SelfRange sobre una intersección de conjuntos:

```
CTimeSet cts1 = WD(7) * (WD(1) + Hour(9));
CTimeSet cts3 = SelfRange(cts1, 1, 5);
resultado:
WD(1) * (WD(2) + Hour(10)) +
WD(2) * (WD(3) + Hour(11)) +
WD(3) * (WD(4) + Hour(12)) +
WD(4) * (WD(5) + Hour(13)) +
WD(5) * (WD(6) + Hour(14))
```

▷ SelfRange sobre una diferencia de conjuntos (I):

```
CTimeSet prCts1 = Inside(y2005) - Inside(y2005m12);
CTimeSet cts1 = SelfRange(prCts1, 1,2);
resultado: (y2006 - y2006m1) + y2007
```

▷SelfRange sobre una diferencia de conjuntos (II):

```
CTimeSet prCts2 = Year(2004) - Month(11);
CTimeSet cts2 = SelfRange(prCts2, 2,3);
resultado: (y2006 - y2006m1) + (y2007 - y2007m2)
```

Periodic(CTime D, Real P [, CTimeSet units = CTSAll])

Devuelve el conjunto temporal periódico, compuesto por un `CTime` y todas sus translaciones a un número de ITs múltiplo de `P` dentro del fechado de unidades dado.

SelfPeriodic(CTimeSet ctms, Real p)

Devuelve el conjunto temporal periódico p unidades en la granularidad propia.
Ejemplo:
El conjunto temporal que representa los instantes de tiempo cada 4 segundos puede expresarse así:

```
CTimeSet jede4sec = SelfPeriodic(Second(0), 4);
```

2.2.5. Otras operaciones

TmsOfSer(CSeries ser)

Devuelve el conjunto temporal de todas las fechas del fechado de una serie en las que ésta no se anula.

Capítulo 3

API de CSeries

Índice alfabético