



intervalul [0,50]. Pentru aceasta, în caseta de editare **Minimum Value**: vom înscrie valoarea 0, iar în caseta **Maximum Value**: valoarea 50;

Pentru a face funcțional tandemul spin-casetă de editare, vom utiliza următoarele metode puse la dispoziție de clasa **CSpinButtonCtrl**, care mapează controlul spin:

- **CWnd\* SetBuddy(CWnd\* pWndBuddy)** – funcția asociază un control pentru care să fie modificate valorile. Primește ca parametru pointerul **pWndBuddy** către controlul asociat și returnează un pointer spre un alt control, care fusese asociat anterior, sau **NULL** dacă controlul spin nu a avut un alt control asociat anterior;
- **CWnd\* GetBuddy()** – returnează un pointer spre controlul asociat controlului spin;
- **int SetPos(int nPos)** – setează noua valoare **nPos** pentru control. Această valoare trebuie să fie în interiorul intervalului în care controlul poate lua valori. Funcția returnează vechea valoare care fusese reprezentată de control;
- **int GetPos()** – returnează în octetul mai puțin semnificativ valoarea reprezentată de control;
- **void SetRange(int nLower, int nUpper)** – setează intervalul în care controlul poate modifica valorile, cuprins între **nLower** și **nUpper**;
- **void SetRange32(long nLower, long nUpper)** – setează intervalul în care controlul poate modifica valorile, cuprins între **nLower** și **nUpper**, reprezentate pe 32 de biți;
- **void GetRange(int& nLower, int& nUpper)** – returnează în **nLower** și **nUpper** intervalul în care controlul poate lua valori;
- **void GetRange32(long& nLower, long& nUpper)** – returnează în **nLower** și **nUpper** intervalul în care controlul poate lua valori;

Va trebui acum să asociem controlul spin cu caseta de editare, să marcăm același interval de valori pentru controlul spin cu cel asociat variabilei asociate casetei de editare și să poziționăm de exemplu valoarea pe mijlocul intervalului. Pentru aceasta, să ne reamintim că, ori de câte ori lansăm în execuție un program de tip **Dialog**, constructorul clasei de dialog va lansa în execuție funcția **OnInitDialog()**. În această funcție va trebui să facem și noi inițializările:

```
BOOL CWiz4Dlg::OnInitDialog()
{
    ...
    // TODO: Add extra initialization here
    CEdit* pValoare=(CEdit*)GetDlgItem(IDC_VALOARE);
    CSpinButtonCtrl* pSpin=
        (CSpinButtonCtrl*)GetDlgItem(IDC_SPIN_VALOARE);
    pSpin->SetBuddy(pValoare);
    pSpin->SetRange(0,50);
    pSpin->SetPos(25);
    return TRUE; // return TRUE unless you ...
}
```

## 7.2 Controale indicator de evoluție (Progress)

Controalele indicator de evoluție sunt utilizate uzual pentru a evidenția stadiul de desfășurare a unei acțiuni de durată. Controalele de tip indicator de evoluție apar în general sub forma unui cursor, care se deplasează sau umple interiorul unei casete.

Vom adăuga casetei de dialog un control de tip **Progress** (fig. 7.2).

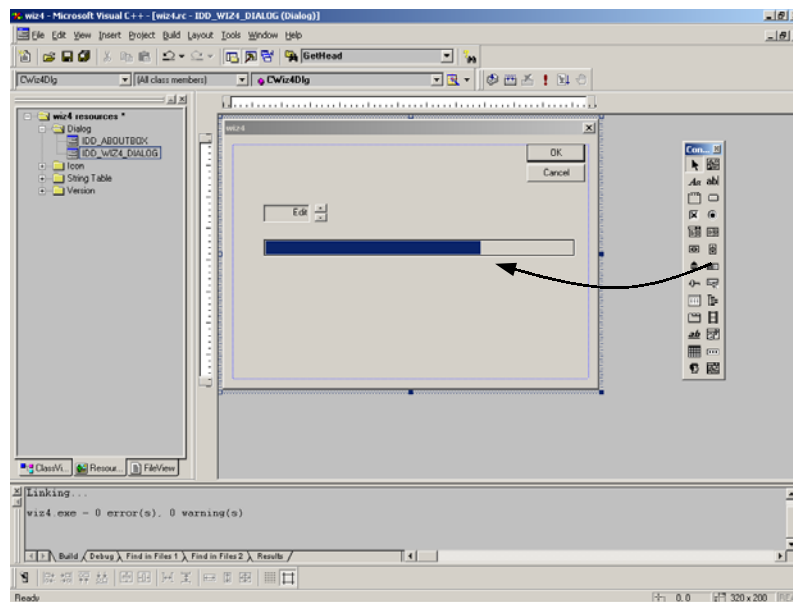


Figura 7.2 Am adăugat un indicator de evoluție

Indicatorul de evoluție poate fi afișat utilizând mai multe stiluri, disponibile la eticheta **Styles** de la **Properties**:

- **Border** – selecția acestei opțiuni de stil are ca efect încadrarea controlului cu un chenar subțire, negru;
- **Vertical** – selecția acestei opțiuni de stil are ca efect desenarea controlului pe verticală;
- **Smooth** – selecția acestei opțiuni de stil are ca efect desenarea în interiorul controlului a unei bare continue, nu formată din blocuri.

Vom modifica identificatorul controlului de evoluție în `IDC_PROGRESS_VALOARE` și vom seta stilurile **Border** și **Smooth**. De asemenea, vom asocia pentru control în **ClassWizard** variabila `m_prValoare`, de categorie **Control** și clasă **CProgressCtrl**. Câteva din metodele puse la dispoziția utilizatorului de această clasă sunt:

- `int StepIt()` – avansează cu un pas controlul;
- `int SetStep(int nStep)` – setează pasul de avans la valoarea `nStep`. Returnează vechea valoare a pasului;
- `int SetPos(int nPos)` – setează poziția absolută a indicatorului de evoluție la `nPos`. Returnează vechea poziție;
- `int OffsetPos(int nPos)` – setează poziția relativă față de vechea poziție pentru indicatorul de evoluție la `nPos`. Returnează vechea poziție;
- `int GetPos()` – returnează poziția controlului de evoluție;
- `void SetRange(int nLower, int nUpper)` – setează intervalul în care controlul poate modifica valorile, cuprins între `nLower` și `nUpper`;
- `void SetRange32(long nLower, long nUpper)` – setează intervalul în care controlul poate modifica valorile, cuprins între `nLower` și `nUpper`, reprezentate pe 32 de biți;
- `void GetRange(int& nLower, int& nUpper)` – returnează în `nLower` și `nUpper` intervalul în care controlul poate lua valori;

- **void GetRange32(long& nLower, long& nUpper)** – returnează în nLower și nUpper intervalul în care controlul poate lua valori;

Va trebui acum să stabilim intervalul de evoluție, pasul și poziția inițială pentru indicatorul de evoluție, în concordanță cu celelalte controale. Acest lucru îl facem tot în funcția `OnInitDialog()`:

```
BOOL CWiz4Dlg::OnInitDialog()
{
    ...
    pSpin->SetPos(25);
    m_prValoare.SetRange(0,50);
    m_prValoare.SetStep(1);
    m_prValoare.SetPos(25);
    return TRUE; // return TRUE unless you ...
}
```

Dacă dorim de exemplu ca să reprezentăm prin poziția controlului de evoluție valoarea înscrisă în caseta de dialog, va trebui să reactualizăm poziția acestuia la fiecare modificare a valorii. Să ne reamintim că o modificare a conținutului unei casete de editare, generează mesajul `EN_CHANGE`. Acestui mesaj va trebui să îi asociem funcția în care se actualizează poziția indicatorului de evoluție:

```
void CWiz4Dlg::OnChangeValoare()
{
    ...
    // TODO: Add your control notification handler code here
    UpdateData();
    m_prValoare.SetPos(m_nValoare);
}
```

Funcția înscrisă întâi valoarea din caseta de editare în variabila asociată, după care modifică poziția controlului de evoluție la valoarea respectivă.

### 7.3 Control glisor (Slider)

Controlul glisor permite utilizatorului să stabilească o valoare, prin deplasarea unui cursor. Deplasarea se poate realiza sub controlul mouse-ului, sau prin intermediul tastelor săgeată, dacă este selectat controlul glisor.

Să adăugăm casetei de dialog un control glisor (fig. 7.3), căruia să îi atribuim identificatorul `IDC_SLIDER_VALOARE`.

Controlul glisor poate fi afișat utilizând mai multe stiluri, disponibile la eticheta **Styles** de la **Properties**:

- **Orientation** – specifică orientarea orizontală sau verticală a controlului;
- **Point** – stabilește tipul cursorului:
  - **Both** – cursorul este dreptunghiular;
  - **Top/Left** – cursorul are un vârf îndreptat în sus în cazul unui cursor orizontal, respectiv înspre stânga, în cazul unui cursor vertical;
  - **Bottom/Right** – cursorul are un vârf îndreptat în jos, dacă cursorul este orizontal, respectiv înspre dreapta, dacă cursorul este vertical;

- **Tick marks** – validarea acestei opțiuni de stil va avea ca efect afișarea de mici diviziuni pe direcția cursorului, situate în direcția orientării cursorului;
- **Auto ticks** - validarea acestei opțiuni de stil are ca efect plasarea diviziunilor de-a lungul intervalului, astfel încât să corespundă valorii incrementale curente;
- **Enable selection** – validarea acestei de stil opțiuni are ca efect adăugarea unei bare albe, care permite programului să afișeze un interval de selecție prin intermediul unor mici triunghiuri;
- **Border** – validarea acestei opțiuni de stil va afișa un chenar în jurul controlului.

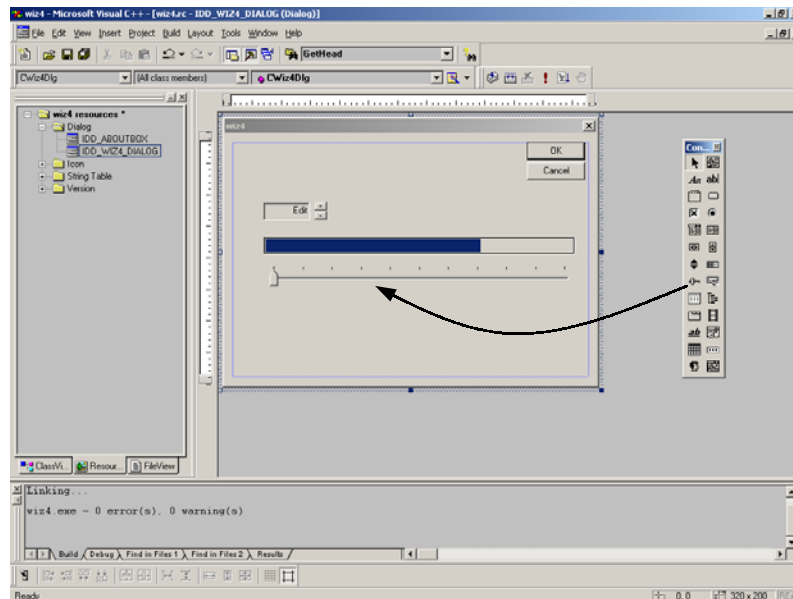


Figura 7.3 Am adăugat și un glisor

Vom seta stilurile **Orientation: Horizontal**; **Point: Top/Left**; **Tick marks** și **Auto ticks** pentru controlul glisor. De asemenea, vom asocia pentru control în **ClassWizard** variabila `m_slValoare`, de categorie **Control** și clasă **CSliderCtrl**. Câteva din metodele puse la dispoziția utilizatorului de această clasă sunt:

- **void SetPos(int nPos)** – setează poziția absolută a glisorului la `nPos`;
- **int GetPos()** – returnează poziția glisorului;
- **void SetRange(int nMin, int nMax, BOOL bRedraw = FALSE)** – setează intervalul în care controlul poate modifica valorile, cuprins între `nMin` și `nMax`. Dacă `bRedraw` este **TRUE** (implicit) glisorul este redesenat după fiecare modificare;
- **void GetRange(int& nMin, int& nMax)** – returnează în `nMin` și `nMax` intervalul în care glisorul poate lua valori;
- **void SetTicFreq(int nFreq)** – setează frecvența `nFreq` de apariție a marcajelor bară verticală pentru glisor;

Celelalte metode furnizate de clasa **CSliderCtrl** le puteți găsi în MSDN. Va trebui acum să inițializăm și glisorul. O vom face tot în funcția `OnInitDialog()`:

```

BOOL CWiz4Dlg::OnInitDialog()
{
    ...
    m_prValoare.SetPos(25);
    m_slValoare.SetRange(0,50);
}

```

```

    m_slValoare.SetTicFreq(1);
    m_slValoare.SetPos(25);
    return TRUE; // return TRUE unless you ...
}

```

Deoarece dorim ca la modificarea valorii înscrise în caseta de editare și glisorul să se deplaseze, vom modifica și funcția `OnChangeValoare()`:

```

void CWiz4Dlg::OnChangeValoare()
{
    ...
    m_prValoare.SetPos(m_nValoare);
    m_slValoare.SetPos(m_nValoare);
}

```

De fiecare dată când poziția glisorului este modificată, controlul glisor trimite un mesaj corespunzător ferestrei care îl conține. Mesajele generate la modificarea poziției glisorului sunt `WM_HSCROLL` corespunzătoare poziției orizontale și `WM_VSCROLL`, corespunzătoare respectiv poziției verticale a glisorului. Crearea de funcții asociate acestor mesaje se face urmând secvența cunoscută din **Class Wizard**: la eticheta **Message Maps**, în caseta combinată **Class name**: se alege clasa `CWiz4Dlg`, în caseta **Object IDs**: vom selecta `CWiz4Dlg` și în caseta **Messages**: vom alege `WM_HSCROLL`. Apoi vom apăsa butoanele **Add Function...** și **Edit Code**. **ClassWizard** va declara și defini funcția de mai jos:

```

void CWiz4Dlg::OnHScroll(UINT nSBCode, UINT nPos,
    CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}

```

Funcția primește următorii parametri:

- `nSBCode` este un parametru care arată tipul operației de derulare efectuate de către utilizator. Această operație depinde de modul în care a fost utilizat controlul și de orientarea acestuia. Valorile posibile pentru acest parametru sunt:

**Tabelul 7.1 (OnVScroll)**

Valoare	Semnificație
<code>SB_BOTTOM</code>	Deplasare în partea de jos a controlului;
<code>SB_ENDSCROLL</code>	Terminare deplasare;
<code>SB_LINEDOWN</code>	Deplasare o linie în jos;
<code>SB_LINEUP</code>	Deplasare o linie în sus;
<code>SB_PAGEDOWN</code>	Deplasare o pagină în jos;
<code>SB_PAGEUP</code>	Deplasare o pagină în sus;
<code>SB_THUMBPOSITION</code>	Deplasare la poziția absolută, indicată de <code>nPos</code> și eliberarea butonului mouse-lui;
<code>SB_THUMBTRACK</code>	Tragerea controlului într-o anumită poziție. Această poziție este apoi regăsită în <code>nPos</code> ;
<code>SB_TOP</code>	Deplasare în partea de jos a controlului;

**Tabelul 7.2 (OnHScroll)**

Valoare	Semnificație
SB_LEFT	Deplasare în partea stângă a controlului;
SB_ENDSCROLL	Terminare deplasare;
SB_LINELEFT	Deplasare o linie în stânga;
SB_LINERIGHT	Deplasare o linie în dreapta;
SB_PAGELEFT	Deplasare o pagină în stânga;
SB_PAGERIGHT	Deplasare o pagină în dreapta;
SB_THUMBPOSITION	Deplasare la poziția absolută, indicată de nPos și eliberarea butonului mouse-lui;
SB_THUMBTRACK	Tragerea controlului într-o anumită poziție. Această poziție este apoi regăsită în nPos;
SB_RIGHT	Deplasare în partea dreaptă a controlului;

- nPos va fi actualizat cu valoarea curentă a poziției glisorului;
- pScrollBar este un pointer utilizat pentru identificarea controlului care a generat mesajul;

Să completăm proiectul, astfel încât, cursorul glisorului să modifice valoarea înscrisă în caستا de editare și sincron, controlul indicator de evoluție. Va trebui să completăm funcția `OnHScroll()` ca mai jos:

```
void CWiz4Dlg::OnHScroll(UINT nSBCode, UINT nPos,
    CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    m_nValoare=m_slValoare.GetPos();;
    m_prValoare.SetPos(m_nValoare);
    UpdateData(FALSE);
    CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

Nu am folosit valoarea nPos, pentru că aceasta la eliberarea butonului mouse-lui iese din domeniul de vizibilitate și valorile glisorului și implicit casetei de editare devin 0.

## 7.4 Controale de selecție a datei și orei

VC++ 6.0 oferă două controale pentru manipularea datei și timpului, care pot fi văzute în diferite aplicații Microsoft, cum ar fi Microsoft Outlook. Aceste controale sunt calendarul (controlul `MonthCalendar` și controlul selector dată/oră `DateTimePicker`, fig. 7.4). Acesta din urmă, atunci când este folosit pentru alegerea unei date, apare sub forma unei casete combinate care afișează data în format lung sau scurt. Un click pe butonul cu săgeată va afișa un control de tip calendar. La afișarea orei, se poate modifica în sus sau în jos valoarea timpului.

După cum se vede în figură, am inserat două controale, primul având identificatorul `IDC_DATA` afișând data, iar al doilea având identificatorul `IDC_TIME`, și afișând timpul. În plus, sub al doilea control, am inserat un calendar, numit `IDC_CALENDAR`.

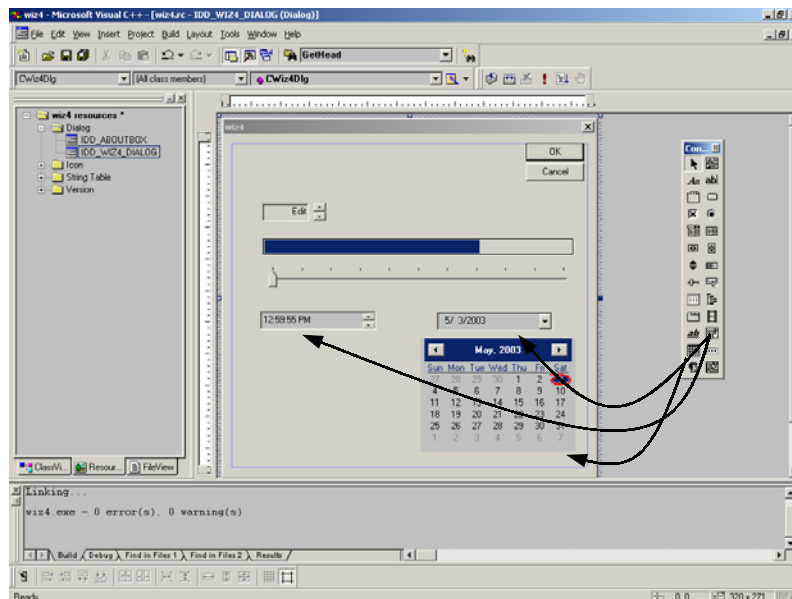


Figura 7.4 Am inserat și controale de timp

### 7.4.1 Controale selector dată/oră

Acest tip de control permite afișarea datei și timpului în diferite formate. Modurile de selecție a datei sau orei sunt stabilite prin intermediul paginii **Styles** de la **Properties**, prin alegerea uneia din cele trei opțiuni afișate de caseta combinată **Format**:

- **Short Date** – setează controlul să afișeze data într-o formă prescurtată, în format american (de exemplu 12 mai 2003 este afișat ca 05/12/03), apăsarea butonului săgeată realizând afișarea unui calendar care permite selectarea mai exactă a datei;
- **Long Date** – setează controlul să afișeze data calendaristică într-o formă detaliată (de ex. Monday, May 12, 2003), apăsarea butonului săgeată realizând afișarea unui calendar care permite selectarea mai exactă a datei;
- **Time** – determină afișarea orei (de ex. 8:49:87 AM) și a unui control de modificare incrementală în ambele sensuri, care permite ajustarea orelor, minutelor, secundelor sau sufixului **AM/PM** din caseta orară;

Se pot stabili de asemenea următoarele opțiuni de stil, pe baza casetelor de validare:

- **Right Align** – selectarea acestei opțiuni de stil are ca efect afișarea calendarului la apăsarea butonului săgeată în dreapta controlului. Pentru opțiunea neselectată, calendarul va fi afișat sub control;
- **Use Spin Control** – selectarea acestei opțiuni de stil are ca efect afișarea în locul butonului săgeată a unui control incremental. Data se poate modifica în sus sau jos, în funcție de săgeata pe care se apasă;
- **Show None** – selectarea acestei opțiuni de stil are ca efect afișarea în stânga selectorului a unei casete de validare, care permite utilizatorului să precizeze că nu este selectată nici o dată și nici o oră atunci când caseta de validare a selectorului nu este bifată;



- **Allow Edit** – selectarea acestei opțiuni de stil permite utilizatorului să facă diferite validări și modificări prin program, pe măsură ce editează conținutul controlului, prin tratarea mesajului `DTN_USERSTRING` trimis de către control.

Dacă la definirea unei variabile mapate peste control, se alege categoria **Value**, caseta **Variable type** va oferi posibilitatea alegerii unei variabile din următoarele două clase, echivalente ca și metode, clase ce încapsulează obiecte ce reprezintă timpul: **CTime** și **COleDateTime**. Este în general de preferat utilizarea celei de a doua clase, pentru că ea poate fi utilizată mai ușor în interacțiunea cu alte programe, fiind recunoscută și de alte limbaje implementate de Microsoft.

Dacă se alege o variabilă de categorie **Control**, variabila va referi un obiect din clasa **CDateTimeCtrl**, care încapsulează acest tip de control. De fapt, pot fi mapate două variabile peste același control, una din clasa **CDateTimeCtrl**, oferind un control mai bun și una din clasa **COleDateTime**, care oferă un acces mai rapid.

Să asociem pentru cele 2 controale, în ordine, variabilele de categorie **Control**, numite respectiv `m_Data` și `m_Timp`.

Tabelul 7.3

Cod	Descriere
yyy	Afișează complet anul (de ex. '2000')
yy	Afișează ultimele 2 cifre ale anului (de ex. '00');
y	Afișează ultima cifră a anului (de ex. '0');
MMMM	Afișează numele complet a lunii (de ex. 'April');
MMM	Afișează numele din trei litere a lunii (de ex. 'Apr');
MM	Afișează numărul lunii pe două cifre (de ex. '04');
M	Afișează numărul lunii pe una sau două cifre (de ex. '4' sau '11');
dddd	Afișează numele complet al zilei din săptămână (de ex. 'Sunday');
ddd	Afișează numele din trei litere al zilei (de ex. 'Sun');
dd	Afișează numărul zilei din lună pe două cifre (de ex. '01');
d	Afișează numărul zilei pe una sau două cifre (de ex. '4' sau '11');
HH	Afișează ora pe 2 cifre în formatul pe 24 de ore;
hh	Afișează ora pe 2 cifre în formatul pe 12 de ore;
H	Afișează ora pe 1 sau 2 cifre în formatul pe 24 de ore;
h	Afișează ora pe 1 sau 2 cifre în formatul pe 12 de ore;
tt	Afișează indicatorul AM/PM pe două caractere;
t	Afișează doar A/P;
mm	Afișează minutele pe două cifre (de ex. '08' sau '48');
m	Afișează minutele pe una sau două cifre (de ex. '8' sau '48');
ss	Afișează secunde pe două cifre (de ex. '08' sau '48');
s	Afișează secunde pe una sau două cifre (de ex. '8' sau '48');

Clasa **CDateTimeCtrl** ne oferă o serie de metode, dintre care putem aminti:

- **BOOL SetRange(COleDateTime\* pMinRange, COleDateTime\* pMaxRange)** – setează intervalul de timp afișat de control. Returnează **TRUE** în caz de succes. Modul de utilizare al funcției este prezentat mai jos:

```
COleDateTime dtMin(2003, 1, 1, 0, 0, 0);
COleDateTime dtMax(2003, 12, 31, 23, 59, 59);
m_Data.SetRange (&dtMin, &dtMax);
```

- **DWORD GetRange(COleDateTime\* pMinRange, COleDateTime\* pMaxRange)** – returnează în `pMinRange` și `pMaxRange` intervalul de timp afișat de control;

- **BOOL SetTime(COLEDateTime& timeNew)** – setează timpul la valoarea înscrisă în obiectul `timeNew`;
- **BOOL GetTime(COLEDateTime& timeDest)** – returnează timpul în obiectul `timeDest`;
- **BOOL SetFormat(LPCTSTR pstrFormat)** – afișează data și timpul cu formatul specificat de `pstrFormat`. Acest parametru poate lua una din valorile din tabelul 7.3;

Este util să amintim și câteva metode ale clasei `COLEDateTime`:

- **int GetYear()** – returnează anul curent, în intervalul 100-9999;
- **int GetMonth()** – returnează luna curentă, în intervalul 1-12;
- **int GetDay()** – returnează ziua, în intervalul 1-31;
- **int GetHour()** – returnează ora, în intervalul 0-23;
- **int GetMinute()** – returnează minutul curent, în intervalul 0-59;
- **int GetSecond()** – returnează secunda curentă, în intervalul 0-59;
- **int GetDayOfWeek()** – returnează ziua curentă din săptămână, după corespundența 1=Sunday..7=Saturday;
- **int GetDayOfYear()** – returnează ziua curentă, considerată în interiorul unui an, cu 1 ianuarie=1 .. 31 decembrie=365/366;

Să asociem de exemplu apăsării butonului `IDOK`, o funcție care va afișa o casetă de dialog cu data curentă în limba română (dacă nu este setată altă dată implicită). Să ne reamintim, pentru adăugarea acestei funcții este suficient să facem dublu click pe acest buton. Funcția va avea implementarea:

```
void CWiz4Dlg::OnOK()
{
    // TODO: Add extra validation here
    COLEDateTime dtDataCurenta;
    CString strDataCurenta, strTemp;
    m_Data.GetTime(dtDataCurenta);
    strTemp.Format("%d ", dtDataCurenta.GetDay());
    switch (dtDataCurenta.GetDayOfWeek()) {
        case 1: strDataCurenta += "Duminica, "; break;
        case 2: strDataCurenta += "Luni, "; break;
        case 3: strDataCurenta += "Marti, "; break;
        case 4: strDataCurenta += "Miercuri, "; break;
        case 5: strDataCurenta += "Joi, "; break;
        case 6: strDataCurenta += "Vineri, "; break;
        case 7: strDataCurenta += "Sambata, "; break;
    }
    strDataCurenta += strTemp;
    switch (dtDataCurenta.GetMonth()) {
        case 1: strTemp = "Ianuarie, "; break;
        case 2: strTemp = "Februarie, "; break;
        case 3: strTemp = "Martie, "; break;
        case 4: strTemp = "Aprilie, "; break;
        case 5: strTemp = "Mai, "; break;
        case 6: strTemp = "Iunie, "; break;
        case 7: strTemp = "Iulie, "; break;
        case 8: strTemp = "August, "; break;
        case 9: strTemp = "Septembrie, "; break;
        case 10: strTemp = "Octombrie, "; break;
    }
```

```

        case 11: strTemp = "Noiembrie, "; break;
        case 12: strTemp = "Decembrie, "; break;
    }
    strDataCurenta += strTemp;
    strTemp.Format("%d", dtDataCurenta.GetYear());
    strDataCurenta += strTemp;
    MessageBox(strDataCurenta, "Data de azi", MB_ICONEXCLAMATION);
    CDialog::OnOK();
}

```

Când modificăm o componentă a controlului, va fi generat mesajul `DTN_DATETIMECHANGE` pe care îl putem intercepta. Acestui mesaj i se poate adăuga prin intermediul **ClassWizard** o funcție de tratare (în **MessageMaps**, se selectează `IDC_DATA` la **Object IDs**: și respectiv `DTN_DATETIMECHANGE` la **Messages**:. Apoi, se apasă **Add Function** și se acceptă numele implicit pentru funcție). Se propune următoarea implementare:

```

void CWiz4Dlg::OnDatetimechangeData(NMHDR* pNMHDR, LRESULT* pResult)
{
    // TODO: Add your control notification handler code here
    COleDateTime dtSelectata;
    switch(pNMHDR->idFrom)
    {
        case IDC_DATA:
            m_Data.GetTime(dtSelectata);
            SetWindowText(" Data: "+dtSelectata.Format("%#x"));
            break;
        case IDC_TIMP:
            m_Timp.GetTime(dtSelectata);
            SetWindowText(" Timp: "+ dtSelectata.Format("%H:%M:%S"));
            break;
    }
    *pResult = 0;
}

```

Funcția de tratare primește ca prim parametru un pointer la o structură `NMHDR`., întâlnită de altfel și în capitolul trecut, structură care identifică obiectul ce a generat mesajul, declarată ca

```

struct tagNMHDR {
    HWND hwndFrom;
    UINT idFrom;
    UINT code;
} NMHDR;

```

unde

- `hwndFrom` este identificatorul ferestrei ce conține controlul care a trimis mesajul;
- `idFrom` este identificatorul controlului care a trimis mesajul;
- `code` este un cod de control sau notificare;

Prin intermediul acestui pointer putem determina care din controale a trimis mesajul, iar titlul casetei de dialog este schimbat în consecință. Apare totuși o problemă: în acest moment, la modificarea datei, titlul casetei va fi schimbat, dar la modificarea orei, programul nu va reacționa. Pentru a înțelege de ce, să ne reamintim că legătura dintre un eveniment și funcția care îl tratează, este specificată printr-o linie

în harta de mesaje `MESSAGE_MAP`. Adăugarea funcției `OnDatetimechangeData()` ca funcție asociată mesajului `DTN_DATETIMECHANGE` generat de controlul `IDC_DATA` cu `ClassWizard` va adăuga automat în harta de mesaje o linie de forma

```
ON_NOTIFY(DTN_DATETIMECHANGE, IDC_DATA, OnDatetimechangeData)
```

Nu vom adăuga o funcție similară și pentru controlul `IDC_TIMP`, deoarece aceasta ar fi `OnDatetimechangeTimp()`, dar noi dorim ca accesul la ambele controale să fie rezolvat de aceeași funcție. Pentru aceasta, vom asocia manual în harta de mesaje, controlului `IDC_TIME`, la producerea evenimentului `DTN_DATETIMECHANGE`, aceeași funcție care este asociată controlului `IDC_DATA`. Macroul va avea implementarea:

```
BEGIN_MESSAGE_MAP(CWiz4Dlg, CDialog)
...
ON_WM_HSCROLL()
ON_NOTIFY(DTN_DATETIMECHANGE, IDC_DATA, OnDatetimechangeData)
ON_NOTIFY(DTN_DATETIMECHANGE, IDC_TIMP, OnDatetimechangeData)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

Acum, totul funcționează corect.

## 7.4.2 Control de tip calendar

Pentru controlul calendar sunt accesibile următoarele stiluri:

- **Day States** – validarea acestei opțiuni de stil permite specificarea prin program a datei ce va fi afișată în culori inverse;
- **Multi Select** – prin validarea acestei opțiuni de stil, utilizatorul va avea posibilitatea selectării unui interval de timp în interiorul calendarului. În caz contrar, implicit, poate fi selectată o singură dată;
- **No Today Circle** – dacă această opțiune de stil este validată, funda roșie ce apare în jurul date curente nu mai este afișată;
- **No Today** – dacă această opțiune de stil este selectată (implicit), data curentă nu mai este afișată explicit în partea de jos a calendarului;
- **Week Numbers** – la validarea acestei opțiuni de stil, în stânga săptămânii afișate va apare și numărul ei în an;

Clasa care descrie controlul de tip calendar este `CMonthCalCtrl`. Dacă controlului calendar i se asociază o variabilă de categorie `Control`, ea va mapa un obiect al acestei clase. Dacă categoria este `Value`, variabila va mapa un obiect `CTime` sau `COleDateTime`, similar cazului precedent. Fie `m_Calendar` variabila de categorie `Control` asociată controlului calendar. Clasa `CMonthCalCtrl` oferă o serie de metode, printre care amintim:

- **BOOL SetRange(COleDateTime\* pMinRange, COleDateTime\* pMaxRange)** – setează intervalul de timp afișat de control. Returnează `TRUE` în caz de succes;
- **DWORD GetRange(COleDateTime\* pMinRange, COleDateTime\* pMaxRange)** – returnează în `pMinRange` și `pMaxRange` intervalul de timp afișat de control;

- **BOOL SetCurSel(COLEDateTime& refDateTime)** – selectează zona din calendar specificată în *refDateTime*;
- **BOOL GetCurSel(COLEDateTime& refDateTime)** – încarcă obiectul *refDateTime* cu zona din calendar selectată;
- **BOOL SetSelRange(COLEDateTime& pMinRange, COLEDateTime& pMaxRange)** – selectează zona din calendar cuprinsă între *pMinRange* și *pMaxRange*;
- **BOOL GetSelRange(COLEDateTime& pMinRange, COLEDateTime& pMaxRange)** – încarcă în *pMinRange* și *pMaxRange* limitele zonei selectate în calendar;
- **BOOL SetMaxSelCount(int nMax)** – setează la valoarea *nMax* numărul maxim de zile ce pot fi selectate în calendar;
- **void SetToday(COLEDateTime& refDateTime)** – setează data curentă afișată de calendar la data conținută de *refDateTime*;
- **BOOL GetToday(COLEDateTime& refDateTime)** – încarcă obiectul *refDateTime* cu zona data curentă;
- **BOOL SetFirstDayOfWeek(int iDay, int\* lpnOld=NULL)** – setează ziua specificată de *iDay* ca fiind prima zi a săptămânii, în funcție de zona geografică. De exemplu, în SUA, duminică=0, sâmbătă=7, în România luni=0, duminică=7. De exemplu, pentru ca luni să fie prima zi a săptămânii, va apela funcția
- **int GetFirstDayOfWeek(BOOL\* pbLocal=NULL)** – returnează prima zi a săptămânii (luni=0, duminică=7);
- **COLORREF SetColor(int nRegion, COLORREF ref)** – setează culoarea calendarului la valoarea *ref*, conform parametrului *nRegion*. Returnează vechea valoare a culorii. Parametrul *nRegion* poate lua valorile din tabelul 7.4;

Tabelul 7.4

Valoare simbolică	Descriere
MCSC_TEXT	Culoarea pentru afișarea datelor
MCSC_TITLETEXT	Culoarea pentru titlu
MCSC_TITLEBK	Culoarea pentru fundalul titlului
MCSC_TRAILINGTEXT	Culoarea pentru antet și zilele externe
MCSC_MONTHBK	Culoarea de fond pentru lună
MCSC_BACKGROUND	Culoarea pentru fond

Să modificăm de exemplu, funcția `OnInitDialog()` ca mai jos:

```

BOOL CWiz4Dlg::OnInitDialog()
{
    ...
    m_slValoare.SetPos(25);
    m_Calendar.SetColor(MCSC_TITLEBK, RGB(0,255,0));
    return TRUE; // return TRUE unless you ...
}

```

Aceasta va avea ca efect colorarea barei de titlu a calendarului în verde. Pentru definirea culorii, am utilizat funcția API

- **COLORREF RGB(BYTE bRed, BYTE bGreen, BYTE bBlue)** – generează o culoare ca și combinație de roșu, verde și albastru, luate în proporțiile *bRed*, *bGreen* și respectiv *bBlue*;

Să selectăm acum pentru calendar stilul **Multi Select**. Acum vom putea selecta în calendar o perioadă de timp, sau o dată singulară. Dacă selectăm o dată singulară, data selectată poate fi preluată prin intermediul parametrului **COleDateTime** a metodei **GetCurSel()**;

Dacă selecția este multiplă, vom utiliza pentru determinarea intervalului selectat, respectiv pentru selectarea unui nou interval, metodele **GetSelRange()** și **SetSelRange()**. Implicit, numărul maxim de zile permis la o selecție multiplă este 7. Această valoare implicită poate fi modificată prin metoda **SetMaxSelCount()**.

Să modificăm acum **OnInitDialog()** ca mai jos:

```
BOOL CWiz4Dlg::OnInitDialog()
{
    ...
    m_Calendar.SetColor(MCSC_TITLEBK, RGB(0,255,0));
    m_Calendar.SetMaxSelCount(14);
    m_Calendar.SetSelRange(COleDateTime (2003,1,1,0,0,0),
        COleDateTime (2003,12,31,23,59,59));
    return TRUE; // return TRUE unless you ...
}
```

Prin aceasta, am permis selectarea a maximum 14 zile, din timpul anului 2003. Acum, să înlocuim ultimele două linii din funcția **OnOK()** ca mai jos:

```
void CWiz4Dlg::OnOK()
{
    ...
    // strDataCurenta += strTemp;
    // MessageBox(strDataCurenta,"Data de azi",MB_ICONEXCLAMATION);

    COleDateTime dtMin, dtMax;
    m_Calendar.GetSelRange(dtMin, dtMax);
    COleDateTimeSpan dtSpan = dtMax-dtMin;
    strDataCurenta += dtSpan.Format("\n Ati selectat %D zile,");
    AfxMessageBox(strDataCurenta);
    CDialog::OnOK();
}
```

Pentru calculul numărului de zile selectate am utilizat un obiect de clasă **COleDateTimeSpan**, care încapsulează intervale de timp. Din această clasă am utilizat metoda

- **CString Format(LPCTSTR pFormat)** – returnează un șir de caractere care conține data/timpul formate. Formatul de afișare este conținut în șirul de formatare (similar funcției **printf()**) **pFormat** și poate fi:

**Tabelul 7.5**

Format	Descriere
%H	ore, în cadrul zilei curente
%M	minute, în cadrul orei curente
%S	secunde, în cadrul minutului curent
%D	format zecimal, etc

Și în acest caz, controlul trimite mesaje pentru a specifica diferite evenimente. De exemplu, un astfel de mesaj este **MCN\_SELCHANGE**, trimis la modificarea selecției în

cadrul calendarului. Cu **ClassWizard** se poate asocia o funcție acestui mesaj, care se propune să fie completată ca mai jos:

```
void CWiz4Dlg::OnSelchangeCalendar(NMHDR* pNMHDR, LRESULT* pResult)
{
    // TODO: Add your control notification handler code here
    COleDateTime dtMin, dtMax;
    m_Calendar.GetSelRange(dtMin, dtMax);
    COleDateTimeSpan dtSpan = dtMax-dtMin;
    SetWindowText(dtSpan.Format(" Ati selectat %D zile,"));
    *pResult = 0;
}
```

## Întrebări și probleme propuse

1. Implementați și executați toate exemplele propuse în capitolul 7;
2. Anulați opțiunea **Read\_only** pentru caseta de editare. Dacă acum tastăm un număr în interiorul casetei, programul funcționează corect? Explicați.
3. Modificați funcția `OnChangeValoare()` ca mai jos. Ce efect au liniile de cod nou introduse? De ce nu se modifică secunde și la acționarea glisorului? Modificați programul astfel încât și acționarea glisorului să modifice secunde în controlul de timp.

```
void CWiz4Dlg::OnChangeValoare()
{
    ...
    COleDateTime dtData;
    UpdateData();
    m_prValoare.SetPos(m_nValoare);
    m_slValoare.SetPos(m_nValoare);
    m_Timp.GetTime(dtData);
    dtData.SetTime(dtData.GetHour(), dtData.GetMinute(), m_nValoare);
    m_Timp.SetTime(dtData);
}
```

4. Modificați funcțiile `OnDatetimestampchangeData()` și `OnSelchangeCalendar()` ca mai jos. Ce efect au modificările realizate?

```
void CWiz4Dlg::OnDatetimestampchangeData(NMHDR* pNMHDR,
    LRESULT* pResult)
{
    // TODO: Add your control notification handler code here
    COleDateTime dtSelectata;
    switch(pNMHDR->idFrom)
    {
        case IDC_DATA:
            m_Data.GetTime(dtSelectata);
            m_Calendar.SetToday(dtSelectata);
            SetWindowText(" Data: "+dtSelectata.Format("%#x"));
            break;
        ...
    }
    *pResult = 0;
}
```

```
void CWiz4Dlg::OnSelchangeCalendar(NMHDR* pNMHDR,
    LRESULT* pResult)
{
    ...
    m_Calendar.GetSelRange(dtMin, dtMax);
    m_Data.SetTime(dtMin);
    COleDateTimeSpan dtSpan = dtMax-dtMin;
    SetWindowText(dtSpan.Format(" Ati selectat %D zile,"));
    *pResult = 0;
}
```

5. Inserați în caseta de dialog un control glisor cu intervalul de valori cuprins între 1 și 31. Modificați data curentă în luna mai 2003 în cele 2 controale care reprezintă data (selector de dată și calendar) în funcție de poziția glisorului. Modificați poziția glisorului în funcție de data selectată în cele 2 controale de dată.