

6. Controale de tip listă

O mare parte din mesajele afișate de interfața programului este reprezentată de mesajele tip listă. Astfel, pot apare liste de obiecte de inventar, liste de posibilități de transport, directoare sau fișiere. Modalitatea de afișare a elementelor unei liste depinde de semnificația acestora. MFC oferă mai multe tipuri de controale de tip listă, cu propriile caracteristici și propriile stiluri.

Controalele de tip listă sunt oferite în 4 forme: *casete combinate*, *casete cu listă*, *arbori* și respectiv *controale listă*. La adăugarea controalelor de tip listă, este foarte importantă selectarea proprietăților de stil adecvate, o alegere nefericită a acestora putând altera semnificativ aspectul listei.

6.1 Casete combinate (ComboBox)

Un control de tip casetă combinată este conține 3 controale: o casetă de editare, o casetă cu listă și un buton. Casetele combinate realizează afișarea unei liste de opțiuni, permițând selectarea unei singure opțiuni. Elementul selectat va fi întotdeauna vizibil, fiind afișat în partea superioară a controlului.

Există trei tipuri de casete combinate:

- **simple** – îmbină o casetă de editare și o casetă listă. Lista este întotdeauna vizibilă și elementul selectat este afișat în caseta de editare;
- **derulante** – îmbină o casetă de editare cu buton și o casetă cu listă. Caseta cu listă este vizibilă numai la efectuarea unui click pe buton;
- **liste derulante** – îmbină o etichetă statică cu un buton și o casetă cu listă. Este aproximativ similară cu caseta derulantă, dar utilizatorul nu poate introduce date în cadrul controlului;

Tipul casetei se selectează la opțiunea **Styles** a casetei de dialog **Combo Box Properties**.

Pentru adăugarea unei casete combinate, se realizează următoarele:

- se creează proiectul *wiz3*;
- se șterg caseta **TODO** și butonul **Cancel**;
- se deplasează în dreapta jos butonul **OK**;
- se inserează eticheta statică **Director rădăcină** în partea stânga sus a machetei;
- se inserează și se aliniază caseta combinată cu buton (**Combo Box**);
- se denumește caseta combinată `IDC_DIRECTOR_PRINCIPAL`;
- la **Styles->Type** se selectează **Drop list**;

S-a obținut caseta din figura 6.1.

La adăugarea casetei combinate într-o casetă de dialog, este selectată implicit opțiunea de stil **Sort**. Aceasta înseamnă că elementele listei vor fi afișate în ordine alfabetică. În marea majoritate a cazurilor, acest lucru ne va încurca. De aceea e nevoie de eliminarea acestei opțiuni. Asta se face foarte ușor, pur și simplu se anulează selecția opțiunii **Sort** la **Styles**.

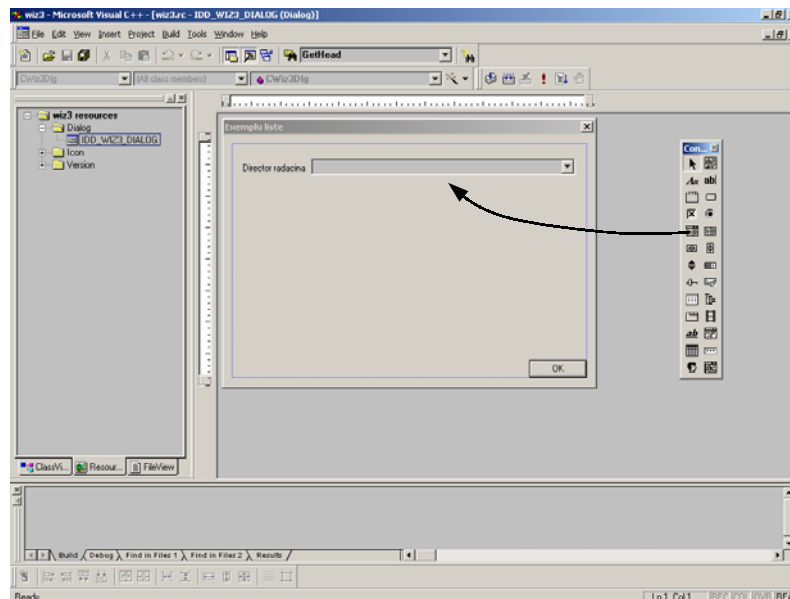


Figura 6.1 Inserăm o casetă combinată

În forma în care este inserată caseta combinată, ea nu va afișa conținutul listei asociate. Va trebui să extindem această listă, pentru a cuprinde mai multe elemente. Acest lucru se face foarte simplu: se face click pe butonul casetei combinate și se trage în jos dreptunghiul de marcare afișat.

Ca să manipulăm caseta combinată în cadrul programului, va trebui să îi mapăm un obiect de aceeași clasă: `CComboBox`. Să ne reamintim cum facem asta:

- lansăm **Class Wizard**;
- selectăm eticheta **Member Variables**;
- în caseta **Class name**: avem grijă să fie selectat `CWiz3Dlg`;
- în caseta **Control IDs**: selectăm `IDC_DIRECTOR_PRINCIPAL`;
- apăsăm butonul **Add Variable...**;
- în caseta **Add Member Variable** selectăm în caseta de editare **Category**: o variabilă de categoria **Control** și verificăm dacă la **Variable type**: este declarată clasa `CComboBox`;
- în caseta **Member variable name**: se introduce `m_cbDirPrinc`;

6.2 Popularea casetelor combinate

O casetă combinată poate fi populată manual, selectând eticheta **Data** din **Properties**, dar nu este prea indicat. În general, caseta combinată se populează prin program, în interiorul unor funcții asociate clasei de dialog. Funcția care realizează popularea inițială a casetei, este apelată în marea majoritate a cazurilor în funcția `OnInitDialog()`. MFC apelează această funcție la deschiderea casetei de dialog, înainte de afișarea ei.

Clasa `CComboBox` conține mai multe metode pentru manipularea casetelor combinate, dintre care cele mai uzuale sunt prezentate mai jos:

Regăsirea fiecărui element în lista asociată casetei combinate se face pe baza unui indice asociat, primul element având indicele 0.

- `int GetCurSel()` – returnează indicele din lista asociată casetei a elementului selectat cu bara luminoasă;

- **int GetCount()** – returnează numărul de elemente din lista asociată casetei combinate;
- **void GetLBText(int nIndex, CString& rString)** – încarcă în șirul *rString* textul de la indexul *nIndex* din lista asociată casetei combinate;
- **int AddString(LPCTSTR lpszString)** – funcția adaugă șirul de caractere *lpszString* în lista asociată casetei combinate și returnează poziția pe care a fost inserat șirul. Dacă proprietatea *Sort* este validată, șirul este inserat pe poziția potrivită, altfel este inserat la coada listei;
- **int DeleteString(UINT nIndex)** – șterge șirul de pe poziția *nIndex* din lista asociată casetei combinate. Returnează numărul de șiruri rămase în listă;
- **void ResetContent()** – șterge toate șirurile de caractere din lista asociată casetei combinate;
- **int SelectString(int nStartAfter, LPCTSTR lpszString)** - caută șirul *lpszString* în lista asociată casetei combinate. Dacă îl găsește, returnează indicele acestuia și îl afișează în casetă. Dacă nu îl găsește, returnează *CB_ERR*. *nStartAfter* reprezintă indicele de la care începe căutarea șirului. Dacă dorim căutarea în întreaga listă asociată casetei combinate, *nStartAfter* va avea valoarea *-1*;

Să facem un exemplu: să construim o funcție *PopulezCombo()*, care realizează popularea inițială a casetei combinate cu numele directorului implicit în care se află programul, al directorului rădăcină de pe discul pe care se află sistemul de operare, al directorului care conține sistemul de operare și al subdirectorului *System* al acestuia. Vom folosi funcțiile:

- **UINT GetWindowsDirectory(LPTSTR lpBuffer, UINT uSize)** – salvează în șirul *lpBuffer* calea spre directorul *Windows*. Calea poate avea maximum *uSize* caractere. Funcția returnează lungimea căii.
- **UINT GetSystemDirectory(LPTSTR lpBuffer, UINT uSize)** – salvează calea spre subdirectorul *System*;
- **DWORD GetCurrentDirectory(DWORD nBufferLength, LPTSTR lpBuffer)** – determină calea spre directorul curent;
- **char *strcpy(char *strDestination, const char *strSource)** - copiază conținutul șirului sursă peste șirul destinație;

Vom utiliza și următoarea constantă:

- **MAX_PATH** – lungimea maximă care o poate lua șirul de caractere care reprezintă calea spre un director. Uzual, este 260;

Pentru popularea listei, se va adăuga la clasa *CWiz3Dlg* funcția membru *void PopulezCombo()*, cu implementarea de mai jos:

```
void CWiz3Dlg::PopulezCombo()
{
    TCHAR TamponCale[MAX_PATH];
    TCHAR TamponCopieCale[MAX_PATH];
    GetWindowsDirectory(TamponCale, MAX_PATH);
    strcpy(TamponCopieCale, TamponCale);
    TamponCopieCale[2]=0;
    m_cbDirPrinc.AddString(TamponCopieCale);
}
```

```

m_cbDirPrinc.AddString(TamponCale);
GetSystemDirectory(TamponCale, MAX_PATH);
m_cbDirPrinc.AddString(TamponCale);
GetCurrentDirectory(MAX_PATH, TamponCale);
m_cbDirPrinc.AddString(TamponCale);
}

```

Funcția citește pe rând, căile spre directorul *Windows*, directorul *System* și directorul curent în șirul *TamponCale*. Acest șir este adăugat apoi prin metoda *AddString()* la obiectul de tip control listă. Pentru determinarea directorului rădăcină, se copiază *szTamp* în *TamponCopieCale*, iar la indexul 2 al acestuia se inserează valoarea 0, care, după cum știm, este terminator de șir. Astfel, *TamponCopieCale* va conține doar *c:*, sau litera corespunzătoare unității logice.

După implementarea acestei funcții, se modifică o linie în *OnInitDialog()* ca mai jos, pentru a apela funcția *PopulezCombo()*:

```

BOOL CWiz7Dlg::OnInitDialog()
{
    ...
    // TODO: Add extra initialization here
    PopulezCombo();
    return TRUE; // return TRUE unless you ...
}

```

Preluarea elementului din lista asociată casetei, selectat la un moment dat, se face prin interceptarea mesajului (generat de caseta combinată) *CBN_SELCHANGE*. Vom asocia cu ajutorul **ClassWizard**, casetei *IDC_MAIN_DIRECTORY* funcția *OnSelChangeMainDirectory()*, care răspunde mesajului *CBN_SELCHANGE* generat de aceasta.

Șirul extras din caseta combinată va fi păstrat la fel ca și în cazurile precedente într-o variabilă de tip *CString*, pe care o vom numi *m_strSelect* și o vom adăuga ca și variabilă membru a clasei *CWiz3Dlg*. Funcția *OnSelChangeMainDirectory()* va avea implementarea:

```

void CWiz7Dlg::OnSelchangeMainDirectory()
{
    // TODO: Add your control notification handler code here
    nIndex = m_cbDirPrinc.GetCurSel();
    if (nIndex != CB_ERR)
    {
        m_cbDirPrinc.GetLBText(nIndex, m_strSelect);
        MessageBox(m_strSelect);
    }
}

```

Funcția încarcă întâi în variabila *nIndex* poziția în lista asociată casetei combinate a șirului de caractere selectat. Pe baza acestui index, șirul este extras și stocat în variabila *m_strSelect*, iar apoi este afișat. Variabila *nIndex* este de asemenea adăugată ca și variabilă membru a clasei *CWiz3Dlg*.

Metodele asociate clasei *CComboBox* utilizate în aceste exemple pot întoarce două mesaje de eroare: *CB_ERR*, ceea ce înseamnă apariția unei erori la apelul metodei, sau *CB_ERRSPACE*, care înseamnă că spațiul disponibil listei asociate casetei combinate este insuficient pentru manipularea șirurilor (la metodele care lucrează cu șiruri ca *AddString()*, *InsertString()*, etc).

Să facem acum ca în caseta combinată să fie afișat implicit directorul rădăcină al discului care conține sistemul de operare. Pentru aceasta, va trebui să modificăm funcția `PopulezCombo()` ca mai jos:

```
void CWiz3Dlg::PopulezCombo()
{
    TCHAR    TamponCale[MAX_PATH];
    ...
    m_cbDirPrinc.AddString(TamponCale);
    nIndex=m_cbDirPrinc.SelectString(-1,TamponCopieCale);
    m_cbDirPrinc.GetLBText(nIndex, m_strSelect);
}
```

6.3 Controale de tip arbore (TreeCtrl)

Controlul arbore este utilizat pentru afișarea și selectarea informațiilor într-un mod ierarhic. Clasa care descrie aceste controale este `CTreeCtrl`. Să adăugăm un control arbore casetei de dialog, ca în figura 6.2:

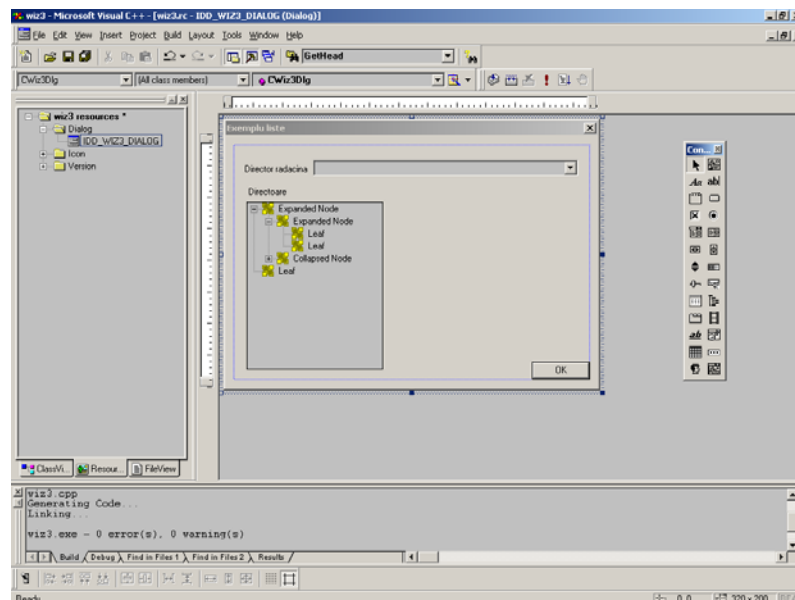


Figura 6.2 Controlul arbore

Pentru inserarea controlului arbore, se execută următoarele:

- se inserează eticheta statică `Directoare`;
- se inserează de pe bara **Controls** controlul arbore;
- i se atribuie identificatorul `IDC_ARBORE_DIRECTOARE`;
- se selectează **Has Buttons**, **Has Lines** și **Lines as Root** la eticheta **Styles** de la **Properties**;

La fel ca și pentru celelalte controale, și pentru manipularea controlului arbore vom avea nevoie fie de un obiect de clasă `CTreeCtrl` mapat peste acesta, fie de un pointer `CTreeCtrl*` obținut cu funcția `GetDlgItem()`. Vom alege prima cale și în **ClassWizard**, vom asocia controlului arbore variabila de categorie **Control** și tip `CTreeCtrl`, numită `m_treeDirectoare`.

Va trebui acum să populăm arborele cu subdirectoarele directorului selectat. Vom aranja în ordine alfabetică în arbore aceste directoare. Deci, pe primul nivel al

arborelui vom pune literele alfabetului și semnul “+” pentru directoarele al căror nume nu începe cu o literă, iar pe al doilea nivel, vom afișa directoarele.

Clasa `CTreeCtrl` include funcțiile pentru manipularea controalelor de tip arbore, dintre care demne de reținut sunt:

- `HTREEITEM InsertItem(LPCTSTR lpszItem, HTREEITEM hParent = TVI_ROOT, HTREEITEM hInsertAfter = TVI_LAST)` – inserează în arbore șirul `lpszItem`;
- `BOOL DeleteItem(HTREEITEM hItem)` – șterge elementul specificat din arbore;
- `BOOL DeleteAllItems()` – șterge toate elementele unui arbore;
- `CString GetItemText(HTREEITEM hItem)` – returnează șirul de caractere conținut de elementul de subarbore `hItem`;

`HTREEITEM` este un tip identificator spre un obiect arbore. Dacă veți dori să lucrați cu acest control, veți observa că MSDN vă prezintă o mulțime de alte metode conținute de clasa `CTreeCtrl`.

Să populăm acum arborele. Vom asocia clasei `CWiz3Dlg` o nouă funcție membru, `void PopulezArbore()`. Apoi, se înlocuiește linia `MessageBox(m_strSelect);` din funcția `CWiz3Dlg::OnSelchangeDirectorPrincipal()` cu `PopulezArbore()`. De asemenea, în funcția `PopulezCombo()` se adaugă aceeași linie:

```
void CWiz3Dlg::PopulezCombo()
{
    ...
    m_cbDirPrinc.GetLBText(nIndex, m_strSelect);
    PopulezArbore();
}
```

Funcția `PopulezArbore()` urmează să aibă implementarea de mai jos:

```
void CWiz3Dlg::PopulezArbore()
{
    m_treeDirectoare.DeleteAllItems();
    HTREEITEM hLitere[27];
    for (int nChar='A'; nChar<='Z'; nChar++)
        hLitere[nChar-'A']=m_treeDirectoare.InsertItem((TCHAR*)&nChar);
    hLitere[26]=m_treeDirectoare.InsertItem("+");

    HANDLE hGasesc;
    WIN32_FIND_DATA dataGasesc;
    BOOL bMaiSunt = TRUE;
    CString strFisier;
    hGasesc=FindFirstFile(m_strSelect+"\\*.*", &dataGasesc);
    while (hGasesc != INVALID_HANDLE_VALUE && bMaiSunt)
    {
        if (dataGasesc.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        {
            int nChar=dataGasesc.cFileName[0];
            if (islower(nChar)) nChar -= 32;
            if (isalpha(nChar)) nChar -= 'A';
            else nChar=26;
            if (strcmp(dataGasesc.cFileName, ".")
                if (strcmp(dataGasesc.cFileName, ".."))
                    m_treeDirectoare.InsertItem(dataGasesc.cFileName,
```

```

        hLitere[nChar]);
    }
    bMaiSunt=FindNextFile(hGasesc, &dataGasesc);
}
FindClose(hGasesc);
}

```

Înainte de a vedea ce am făcut, să înțelegem cum se lucrează cu fișierele. În primul rând, să ne reamintim că orice fișier are asociate o serie de atribute: **Archive**, **ReadOnly**, **System**, **Directory**, etc. Aceste atribute, sunt reprezentate prin valori de tip **DWORD**, care sunt prelucrabile de limbajul VisualC++. Câteva din aceste atribute sunt prezentate în tabelul 6.1:

Tabelul 6.1

Atribut	Valoare	Valoare binară
FILE_ATTRIBUTE_ARCHIVE	32	00100000
FILE_ATTRIBUTE_DIRECTORY	16	00010000
FILE_ATTRIBUTE_HIDDEN	2	00000010
FILE_ATTRIBUTE_READONLY	1	00000001
FILE_ATTRIBUTE_SYSTEM	4	00000100
FILE_ATTRIBUTE_TEMPORARY	256	100000000

Ce putem observa? Aceste valori sunt puteri ale lui 2, adică au în reprezentarea binară doar câte un bit 1, pe o poziție specifică, restul biților fiind 0. În acest fel, unui fișier îi va putea fi asociată o combinație de atribute, combinația făcându-se prin intermediul operatorului **OR** pe bit (**|**). Spre exemplu, un fișier care este **Archive**, **Hidden** și **ReadOnly** va avea atributul **00100011**, obținut ca și expresia **FILE_ATTRIBUTE_ARCHIVE|FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_READONLY**.

API pune la dispoziția programatorilor o serie de funcții pentru manipularea fișierelor. Tehnica utilizată este de a găsi un prim fișier de felul dorit și a-i asocia un identificator. Apoi, pe baza acestui identificator, se vor găsi și celelalte fișiere de același fel din director. Va trebui să utilizăm și noi aceste funcții:

- **HANDLE FindFirstFile(LPCTSTR lpFileName, LPWIN32_FIND_DATA lpFindFileData)** - returnează un identificator pentru primul fișier găsit în calea pointată de **lpFileName** de felul descris în structura **lpFindFileData**;
- **BOOL FindNextFile(HANDLE hFindFile, LPWIN32_FIND_DATA lpFindFileData)** - găsește următorul fișier de felul descris în structura **lpFindFileData** și îi asociază identificatorul **hFindFile** (același cu cel returnat de **FindFirstFile()**). Funcția returnează **TRUE** atâta timp cât mai găsește fișiere de felul dorit. Dacă nu mai există fișiere, returnează **FALSE**.
- **BOOL FindClose(HANDLE hFindFile)** - termină căutarea după identificatorul **hFindFile**.

Observăm că ambele funcții utilizează variabile de tipul structurii **WIN32_FIND_DATA**, declarată ca mai jos:

```

typedef struct _WIN32_FIND_DATA {
    DWORD dwFileAttributes;
    FILETIME ftCreationTime;
    FILETIME ftLastAccessTime;
    FILETIME ftLastWriteTime;
    DWORD nFileSizeHigh;

```

```

DWORD nFileSizeLow;
DWORD dwReserved0;
DWORD dwReserved1;
TCHAR cFileName[ MAX_PATH ];
TCHAR cAlternateFileName[ 14 ];
} WIN32_FIND_DATA

```

Observăm că această structură dă informații despre fișier: attribute, momentul creerii, dimensiune, nume, etc. Aici se rezolvă și misterul numelui fișierelor în Windows: numele întreg al fișierului, inclusiv extensia, este stocat în câmpul `cFileName`. Câmpul `cAlternateFileName` va conține numele alternativ DOS asociat fișierului (poate că ați observat că, de exemplu, fișierul *Fisier cu nume lung.txt* apare în Norton Commander ca *Fisier~1.txt*. Acesta este numele alternativ în format DOS 8.3).

Și acum, să vedem ce facem în funcția `PopulezArbore()`:

- întâi ștergem toate elementele arborelui mapat de `m_treeDirectoare`;
- declarăm apoi un șir de 27 de elemente `HTREEITEM` (26 pentru literele alfabetului + unul pentru directoarele ale căror nume nu începe cu o literă);
- inserăm la indicele corespunzător pe primul nivel al arborelui literele alfabetului: la indicele 'A'-'A'=0 se inserează **A**, la indicele 'B'-'A'=1 se inserează **B**, ș.a.m.d. La indicele 26, se inserează semnul +;
- găsim primul fișier cu numele `m_strSelect+"*.*"`, adică orice fișier, cu orice extensie din directorul descris de calea `m_strSelect`. Cu asta, am încărcat identificatorul `hGasesc` (evident, dacă directorul nu e gol!);
- dacă directorul nu e gol (adică `hGasesc != INVALID_HANDLE_VALUE`) și este `bMaiSunt==TRUE` (implicit), executăm o buclă de program, din care se iese când nu mai există fișiere (`bMaiSunt==FALSE`);
- dacă fișierul găsit are atributul `FILE_ATTRIBUTE_DIRECTORY`, îl inserăm în arbore pe poziția corespunzătoare (după ce i-am transformat primul caracter al numelui în majusculă și am eliminat directoarele . și ..). Aici ar fi câte ceva de spus despre linia

```
if(dataGasesc.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
```

în mod normal, linia ar fi trebuit să fie

```
if(dataGasesc.dwFileAttributes == FILE_ATTRIBUTE_DIRECTORY)
```

În al doilea caz, în arbore ar fi introduse doar fișierele care au strict atributul `FILE_ATTRIBUTE_DIRECTORY`, adică 00010000. În primul caz, pe care l-am preferat, vor fi introduse toate fișierele al căror atribut are bitul 4 pus pe 1, indiferent de valoarea celorlalți biți, adică toate fișierele care au și atributul `FILE_ATTRIBUTE_DIRECTORY`. Această **tehnică de mascare** a tuturor celorlalți biți, este foarte des utilizată în programare. De fapt ce am făcut? Am realizat o operație `AND` pe bit a atributului fișierului cu o **mască**, având 1 doar pe poziția dorită. Rezultatul operației este diferit 0, adică `TRUE`, pentru orice atribut care are același bit ca și masca pus pe 1, indiferent de ce valori au ceilalți biți și respectiv 0 dacă bitul respectiv este 0 (adică `FALSE`). Astfel, vom obține `TRUE` și pentru atributul 0001000, dar și pentru 00010001, sau 10010010, etc.;

- căutăm următorul fișier din director. Dacă mai există fișiere, `bMaiSunt` ia valoarea `TRUE`, altfel ia valoarea `FALSE`;

- în final, distrugem identificatorul asociat fișierelor;

Astfel, am populat controlul de tip arbore.

6.4 Controale de tip casetă cu listă (ListBox)

Casetele cu liste sunt controale care conțin liste de elemente. Spre deosebire de casetele combinate și de controalele tip arbore, care permit numai selecția singulară, casetele cu liste permit următoarele tipuri de selecții:

Tabelul 6.2

Selecție	Descriere
Singulară (Single)	selecția elementelor este mutual exclusivă. Poate fi selectat la un moment dat un singur element, iar selecția lui duce la anularea selecției anterioare;
Multiplă (Multiple)	pot fi selectate mai multe elemente, utilizând mouse-ul în combinație cu Ctrl și Shift ;
Extinsă (Extended)	similar selecției multiple, pot fi selectate mai multe elemente, dar selecția lor se poate face și prin selectare cu mouse-ul, ținând butonul stâng apăsat;
Dezactivată (None)	nu poate fi selectat nici un element;

La fel ca și în cazul casetelor combinate, ordinea alfabetică de afișare a elementelor este selectată implicit. Pentru anularea acestei opțiuni, în caseta **Styles** se deselectează opțiunea **Sort**.

Se propune completarea casetei de dialog asociate proiectului *wiz3* cu o casetă cu listă astfel:

- adăugăm casetei de dialog a proiectului eticheta statică **Fisiere**;

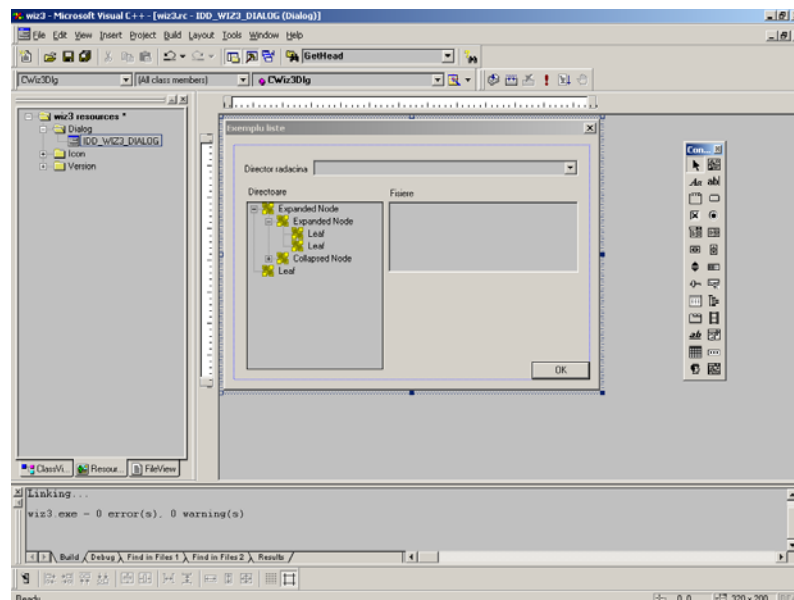


Figura 6.3 Inserarea unei casete cu listă

- alegem din tabelul de controale un control casetă cu listă și îl inserăm în caseta de dialog, iar apoi îi dăm identificatorul `IDC_LISTA_FISIERE`. Vom fi atenți ca la eticheta **Styles** opțiunea **No Integral Height** să fie selectată, iar în caseta combinată **Selection** să fie selectată opțiunea **Extended**. Selecția acestei opțiuni

va avea ca efect afișarea unui număr întreg de linii în casetă. Altfel, ar putea apare linii neafișate integral. Vom obține macheta din fig. 6.3.

- în **ClassWizard**, asociem casetei cu listă variabila de categorie **Control** și clasă **CListBox**, `m_lbFisiere`;

Evident, o dată desenată caseta de dialog și definită variabila, caseta cu listă va trebui populată. Popularea casetei cu listă se face la fel ca și în cazul controalelor anterioare, existând evident deosebirea că în acest caz pot fi selectate simultan mai multe elemente. Metodele oferite de clasa **CListBox** sunt în mare, similare cu cele oferite de clasa **CComboBox**.

Vom adăuga clasei **CWiz3Dlg** funcția membru `void PopulezLista()`, pe care o vom apela în funcția `OnSelchangeDirectorPrincipal()` ca mai jos:

```
void CWiz3Dlg::OnSelchangeDirectorPrincipal()
{
    ...
    PopulezArbore();
    PopulezLista();
}
}
```

Funcția `PopulezLista()` va avea implementarea de mai jos:

```
void CWiz3Dlg::PopulezLista()
{
    m_lbFisiere.ResetContent();

    HANDLE hGasesc;
    WIN32_FIND_DATA dataGasesc;
    BOOL bMaiSunt = TRUE;

    hGasesc=FindFirstFile(m_strSelect+"\\*.*", &dataGasesc);
    while (hGasesc != INVALID_HANDLE_VALUE && bMaiSunt)
    {
        if(dataGasesc.dwFileAttributes == FILE_ATTRIBUTE_ARCHIVE)
            m_lbFisiere.AddString(dataGasesc.cFileName);
        bMaiSunt=FindNextFile(hGasesc, &dataGasesc);
    }
    FindClose(hGasesc);
}
}
```

Va trebui de asemenea să apelăm această funcție și în `PopulezCombo()`, pentru ca o dată cu popularea arborelui de directoare, să fie populată și lista de fișiere:

```
void CWiz3Dlg::PopulezCombo()
{
    ...
    PopulezArbore();
    PopulezLista();
}
}
```

Acțiunile efectuate de utilizator asupra controalelor din caseta cu listă au ca efect transmiterea unor mesaje corespunzătoare. De exemplu, mesajul `LBN_SELCHANGE` este trimis în cazul în care se modifică selecția unuia (sau mai multora) din elementele listei. Modul în care se operează cu o selecție diferă după cum controlul permite sau

nu, selecția multiplă. Cum caseta cu listă permite selecția multiplă, funcția de tratare a mesajului va trebui să extragă numele de fișiere selectate și să le salveze într-o listă. Nu vom crea de această dată o listă simplu înlanțuită, ci vom utiliza un obiect de clasă `CStringList`. Pentru aceasta, vom adăuga clasei `CWiz3D1g` variabila membru `CStringList m_strLista`.

Clasa `CStringList` este o clasă de tip colecție, implementează liste de șiruri de caractere. Fiecare șir din listă poate fi regăsit pe baza unui indice, care din păcate nu este o variabilă întreagă, ci o variabilă de tip `POSITION`. Această variabilă nu poate fi incrementată sau decrementată obișnuit, ci doar prin intermediul metodelor clasei `CStringList`. Câteva din aceste metode sunt:

- `POSITION AddHead(CString newElement)` – adaugă obiectul `newElement` în capul listei și returnează indicele acestuia în listă ;
- `POSITION AddTail(CString newElement)` – adaugă obiectul `newElement` în coada listei și returnează indicele acestuia în listă;
- `int GetCount()` – returnează numărul de elemente din listă;
- `BOOL IsEmpty()` – returnează `TRUE` dacă lista nu mai conține elemente și `FALSE` în caz contrar;
- `CString RemoveHead()` – șterge primul element din listă și îl returnează programului apelant. Lista nu poate fi goală în momentul apelului acestei funcții;
- `CString RemoveTail()` – șterge ultimul element din listă și îl returnează programului apelant. Lista nu poate fi goală în momentul apelului acestei funcții;
- `void RemoveAll()` – șterge toate elementele listei;
- `POSITION GetHeadPosition()` – returnează indicele `POSITION` a primului element din listă. Poate fi utilizat pentru parcurgerea iterativă a listei. Când nu mai există elemente în listă, funcția returnează `NULL`;
- `POSITION GetTailPosition()` – returnează indicele `POSITION` a ultimului element din listă, sau `NULL` dacă lista este goală;
- `CString GetHead()` – returnează primul obiect `CString` din listă;
- `CString GetTail()` – returnează ultimul obiect `CString` din listă;
- `CString GetNext(POSITION rPosition)` – returnează elementul de la poziția curentă din listă, conținută de `rPosition` și trece la poziția următoare;
- `CString GetPrev(POSITION rPosition)` – returnează elementul de la poziția curentă din listă, conținută de `rPosition` și trece la poziția precedentă;
- `CString GetAt(POSITION rPosition)` – returnează elementul de la poziția `rPosition` din listă;

Vom asocia în **ClassWizard** mesajului `LBN_SELCHANGE` generat de controlul `IDC_LISTA_FISIERE` funcția `OnSelchangeListaFisiere()`, cu implementarea de mai jos. Pentru implementarea funcției, va trebui să folosim următoarele metode ale clasei `CListBox`:

- `int GetSelCount()` – returnează numărul de elemente selectate în caseta cu listă, sau `LB_ERR` dacă se încearcă selecția multiplă într-o casetă cu listă cu selecție simplă;

- `int GetSelItems(int nMaxItems, LPINT rgIndex)` – salvează indicii elementelor selectate în listă în șirul de indici `rgIndex`. Numărul maxim de indici salvați este `nMaxItems`. Returnează numărul de indici salvați în `rgIndex`;
- `void GetText(int nIndex, CString rString)` – extrage din lista asociată casetei cu listă șirul de caractere de la indicele `nIndex` și-l salvează în șirul de caractere `rString`;

```
void CWiz3Dlg::OnSelchangeListaFisiere()
{
    // TODO: Add your control notification handler code here
    int nNrSelectate=m_lbFisiere.GetSelCount();
    m_strLista.RemoveAll();
    if (nNrSelectate)
    {
        CString strSelectate;
        LPINT pItem=new int[nNrSelectate];
        m_lbFisiere.GetSelItems(nNrSelectate, pItem);
        for(int i=0;i<nNrSelectate;i++)
        {
            m_lbFisiere.GetText(pItem[i], strSelectate);
            m_strLista.AddTail(strSelectate);
        }
    }
}
```

Funcția începe prin determinarea numărului de elemente selectate în listă. Se crează apoi dinamic un șir de întregi, de dimensiunea numărului de elemente selectate, în care se salvează indicii acestora. Pe baza indicilor, sunt extrase șirurile de caractere din lista asociată casetei cu listă și salvate în obiectul `CStringList` (fig. 6.4). În final, se eliberează memoria de șirul de indici.

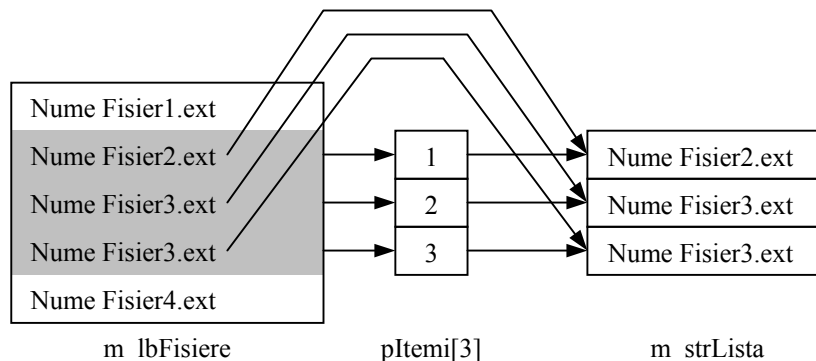


Figura 6.4 Extragerea elementelor selectate în lista

6.5 Să revenim la controlul arbore...

Dacă ne uităm în **ClassWizard** la mesajele generate de controlul arbore, vedem că ele sunt de două feluri: mesaje care au prefixul `NM_` și mesaje care au prefixul `TVM_`. Primele sunt mesaje ce se referă la controlul arbore, iar celelalte se referă la elementele controlului arbore. Să vedem de exemplu, cum putem face ca la alegerea unui nume de director din subarboarele unei litere, caseta cu listă să fie populată de fișierele din acel director.

În primul rând, va trebui să interceptăm mesajul generat de modificarea selecției unui element de subarbore. Acest mesaj este `TVM_SELCHANGED`. Vom asocia în **ClassWizard** acestui mesaj funcția `OnSelchangedArboreDirectoare()`. Putem observa că această funcție a fost declarată ca mai jos:

```
void CWiz3Dlg::OnSelchangedArboreDirectoare(NMHDR* pNMHDR,
    LRESULT* pResult)
{
    // TODO: Add your control notification handler code here
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    *pResult = 0;
}
```

Funcția definește un pointer `pNMTreeView` la o structură `NM_TREEVIEW` declarată ca:

```
struct tagNM_TREEVIEW {
    NMHDR hdr;
    UINT action;
    TVITEM itemOld;
    TVITEM itemNew;
    POINT ptDrag;
} NM_TREEVIEW;
```

Un rol important pentru noi au două din câmpurile acestei structuri:

- `UINT action` – specifică tipul acțiunii care a dus la schimbarea selecției elementului din subarbore. Poate avea valorile:
 - `TVC_BYKEYBOARD` – selecția a fost schimbată la apăsarea unei taste;
 - `TVC_BYMOUSE` – selecția a fost schimbată cu mouse-ul;
 - `TVC_UNKNOWN` – selecția a fost schimbată din cauze necunoscute;
- `TVITEM itemNew` – o structură care conține informații despre elementul din subarbore nou selectat. Structura `TVITEM` este declarată ca și

```
struct tagTVITEM{
    UINT mask;
    HTREEITEM hItem;
    UINT state;
    UINT stateMask;
    LPTSTR pszText;
    int cchTextMax;
    int iImage;
    int iSelectedImage;
    int cChildren;
    LPARAM lParam;
} TVITEM;
```

Informații detaliate despre această structură pot fi găsite în MSDN. Pe noi ne interesează câmpul `HTREEITEM hItem`, pentru că acesta pune în evidență ce element de subarbore a fost selectat.

Vom implementa funcția ca mai jos:

```
void CWiz3Dlg::OnSelchangedArboreDirectoare(NMHDR* pNMHDR,
    LRESULT* pResult)
{
    // TODO: Add your control notification handler code here
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    HTREEITEM it=pNMTreeView->itemNew.hItem;
```

```

CString dir;
dir="\\\\"+m_treeDirectoare.GetItemText(it);
if (pNMTreeView->action==TVC_BYMOUSE)
{
    m_strSelect+=dir;
    PopulezLista();
}
*pResult = 0;
}

```

Întâi am extras în variabila `it`, din structurile asociate elementului de arbore curent, indentificatorul acestuia. Apoi, am format șirul `dir` din numele acestui element de subarbore. Dacă modificarea selecției a fost făcută cu mouse-ul, am adăugat numele acestui director la calea stocată în variabila `m_strSelect` și am populat lista. Să ne reamintim că această cale este actualizată la selecția unui director în caseta combinată.

În acest raționament, apare o mică problemă. Prin adăugarea numelui subdirectorului la calea `m_strSelect`, aceasta este modificată. O nouă schimbare a selecției va adăuga un nou nume de subdirector la această cale, dar noul nume ar trebui adăugat la calea inițială, fiind tot un subdirector al directorului selectat în caseta combinată. Va trebui deci să restaurăm vechea valoare a șirului `m_strSelect`. Pentru aceasta, vom adăuga clasei `CWiz3Dlg` o nouă variabilă membru, `CString m_strCopie`. Vom face în funcții modificările de mai jos:

```

void CWiz3Dlg::OnSelchangeDirectorPrincipal()
{
    ...
    m_cbDirPrinc.GetLBText(nIndex, m_strSelect);
    m_strCopie=m_strSelect;
    PopulezArbore();
    ...
}

void CWiz3Dlg::OnSelchangedArboreDirectoare(NMHDR* pNMHDR,
    LRESULT* pResult)
{
    ...
    CString dir;
    m_strSelect=m_strCopie;
    ...
}

```

6.6 Controale de tip listă (List Control)

Controalele listă sunt cele mai complexe, putând afișa atât pictograme cât și etichete asociate și putând opera în unul din cele 4 moduri de mai jos:

Tabelul 6.3

Mod	Descriere
Pictograme (Icon)	Afișează pictograme mari, de 32x32 pixeli, cu etichete plasate sub fiecare pictogramă. Elementele sunt plasate de la stânga la dreapta și de sus în jos;
Pictograme mici (Small Icon)	Afișează pictograme mici, de 16x16 pixeli, cu etichete plasate la stânga fiecărei pictograme. Elementele sunt plasate de la stânga la dreapta și de sus în jos;
Listă (List)	Afișarea este similară modului cu pictograme mici, dar elementele sunt plasate de sus în jos și apoi de la stânga la dreapta;
Raport (Report)	Afișează informațiile prin intermediul unor coloane din antet;

În cadrul proiectului, vom utiliza controlul listă pentru afișarea de detalii cu privire la fișierele selectate. Pentru aceasta, vom insera în machetă un control listă, cu identificatorul `IDC_DETALII`, și având selectat în caseta combinată **Styles** din cadrul etichetei **View** tipul **Report** (fig. 6.5) Apoi, prin intermediul **ClassWizard**, vom asocia controlului variabila `m_lcDetalii`, de categorie **Control** și clasă `CListCtrl`.

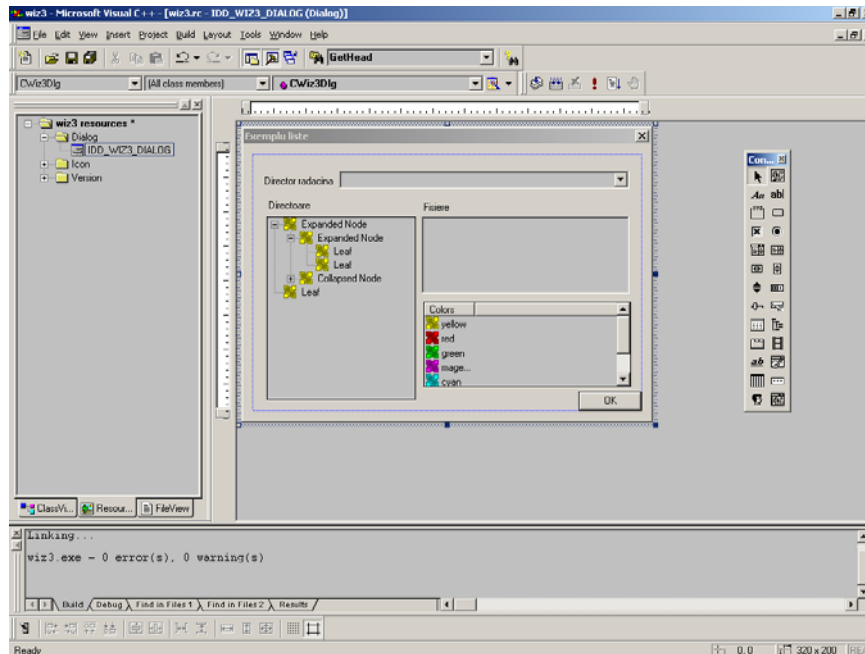


Figura 6.5. Am adăugat control listă

Clasa `CListCtrl` oferă o serie de metode pentru manipularea obiectelor. Câteva dintre acestea sunt prezentate în cele ce urmează:

- `int InsertColumn(int nCol, LPCTSTR lpszColumnHeading, int nFormat=LVCFMT_LEFT, int nWidth=-1, int nSubItem=-1)` – inserează o coloană în controlul de tip listă. Coloana va fi inserată pe poziția `nCol` (prima coloană va avea indicele 0), având titlul `lpszColumnHeading`, textul din coloană fiind aliniat conform valorii `nFormat`, iar coloana are dimensiunea în pixeli dată de `nWidth`. Valoarea -1 pentru acest parametru specifică faptul că dimensiunea coloanei nu este determinată. Parametrul `nFormat` poate lua valorile:
 - `LVCFMT_LEFT` – textul este aliniat spre stânga;
 - `LVCFMT_RIGHT` – textul este aliniat spre dreapta;
 - `LVCFMT_CENTER` – textul este centrat;

Funcția returnează indicele coloanei adăugate, sau -1 în caz de insucces;

- `BOOL DeleteColumn(int nCol)` – șterge coloana de indice `nCol`. Returnează `TRUE` în caz de succes;
- `int InsertItem(int nIndex, LPCTSTR lpszItem)` – inserează elementul de indice `nIndex` (linia `nIndex` în listă) conținând textul `lpszItem`. Returnează indicele noului element introdus în listă în caz de succes, sau -1 în caz de eșec;
- `BOOL SetItemText(int nIndex, int nSubItem, LPCTSTR lpszText)` – inserează textul `lpszText` în linia `nIndex` și coloana `nSubItem` în listă. Returnează `TRUE` în caz de succes;
- `BOOL DeleteAllItems()` – șterge toate elementele controlului listă;

Pentru început, va trebui să formatăm controlul listă. Vom utiliza un control cu 3 coloane. Pentru aceasta, vom adăuga funcției `OnInitDialog()` liniile de mai jos:

```
BOOL CWiz3Dlg::OnInitDialog()
{
    ...
    PopulezCombo();
    m_lcDetalii.InsertColumn(0,"Nume Fisier", LVCFMT_LEFT, 120);
    m_lcDetalii.InsertColumn(1,"Atribute", LVCFMT_CENTER, 120);
    m_lcDetalii.InsertColumn(2,"Dimensiune Kb", LVCFMT_RIGHT, 100);
    return TRUE; // return TRUE unless you ...
}
```

Acum, va trebui să implementăm funcția care populează controlul listă. Vom asocia clasei `CWiz3Dlg` funcția membru `PopulezControlLista()`, cu implementarea de mai jos:

```
void CWiz3Dlg::PopulezControlLista()
{
    m_lcDetalii.DeleteAllItems();
    POSITION Pos;
    int nItem;
    CString strSelectate;
    CString strGasite;
    HANDLE hGasesc;
    WIN32_FIND_DATA dataGasesc;
    for (Pos=m_strLista.GetHeadPosition(); Pos != NULL;)
    {
        strSelectate=m_strLista.GetAt(Pos);
        nItem=m_lcDetalii.InsertItem(0, strSelectate);
        strGasite=strSelectate;
        hGasesc=FindFirstFile(strGasite, &dataGasesc);

        double nDimensiuneFisier=0;
        if (dataGasesc.dwFileAttributes)
            nDimensiuneFisier = (double)(dataGasesc.nFileSizeHigh*MAXWORD
            +dataGasesc.nFileSizeLow)/1024;
        strSelectate.Format("%6.3f", nDimensiuneFisier);
        m_lcDetalii.SetItemText(nItem,1,
            FormezAtribute(dataGasesc.dwFileAttributes));
        m_lcDetalii.SetItemText(nItem,2,strSelectate);
        FindClose(hGasesc);
        m_strLista.GetNext(Pos);
    }
}
```

Funcția parcurge lista de șiruri de caractere poziție cu poziție (până când `pos==NULL!`). Pentru fiecare nume de fișier extras este creat un identificator, cu funcția `FindFirst()`, pe baza căruia sunt extrase informațiile privind dimensiunea fișierului și respectiv atributul asociat. Pentru transformarea atributului din format `DWORD` în șir de caractere, este utilizată funcția `FormezAtribute()`. Dimensiunea fișierului și șirul de atribute este apoi inserat în controlul listă pe linia corespunzătoare, în coloanele 1 și 2.

Va trebui acum să adăugăm funcția `CString FormezAtribute(DWORD Atr)` ca funcție membru a clasei `CWiz3Dlg`:


```

CString CWiz3Dlg::FormezAtribute(DWORD Atr)
{
    CString temp;
    if (Atr&1) temp+="RO ";
    if (Atr&2) temp+="Hy ";
    if (Atr&4) temp+="Sy ";
    if (Atr&32) temp+="Arc ";
    if (Atr&256) temp+="Tmp";
    return temp;
}

```

Pentru ca să putem popula controlul listă, funcția `PopulezControlLista()` va trebui apelată în momentul în care selecția fișierelor este terminată. Va trebui deci să modificăm funcția `OnSelchangeListaFisiere()` ca mai jos:

```

void CWiz3Dlg::OnSelchangeListaFisiere()
{
    ...
    m_strLista.AddTail(strSelectate);
}

PopulezControlLista();
}

```

6.7 Exemplu final. Programarea unui explorer (facultativ)

Ca exemplu final, se propune crearea unui program explorer, care să afișeze discul logic selectat, subdirectoarele din directorul selectat, fișierele din directorul selectat, fiind prezentate numele, dimensiunea și data ultimei înregistrări, precum și calea până la directorul curent. Intrarea într-un director se face prin dublu click asupra lui. Un dublu click asupra directorului “..”, are ca efect reîntoarcerea în directorul părinte. Macheta browserului este cea din fig. 6.6. În exemplu se presupune că proiectul este denumit *Explorer*.

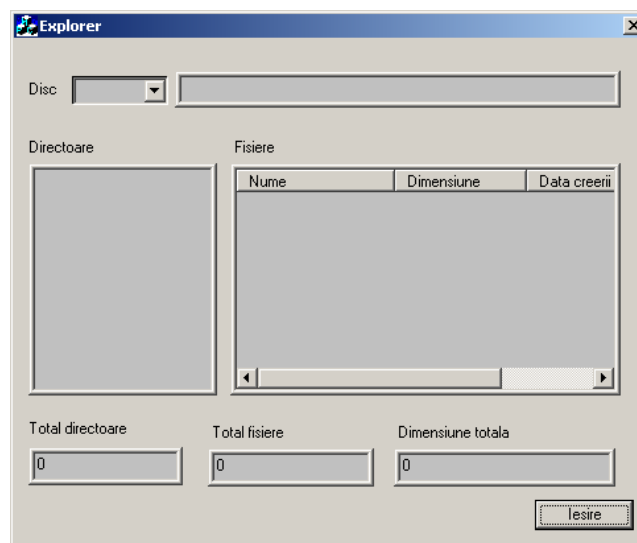


Figura 6.6. Macheta exemplului final

Pe lângă etichetele statice, sunt utilizate următoarele controale:

- o casetă combinată, denumită `IDC_DISC` (pusă în evidență de eticheta `Disc`), fiind de tipul `DropDown`, iar la `Extended Styles` având selectate opțiunile `Client`

Edge și **Modal Frame**. Aceste 2 opțiuni vor fi selectate pentru toate controalele de tip listă și casetă de editare. Casetă are asociată variabila `m_cbDisc` de categorie **Control** și tip **CComboBox**;

- o caseta cu listă, cu identificatorul `IDC_DIRECTOR` (marcată **Director**), permițând selecție de tip **Single** și având asociată variabila `m_lbDirector` de categorie **Control** și tip **CListBox**;
- un control listă, cu identificatoerul `IDC_FISIERE` (marcat **Fisiere**), de tip **Report**, având asociată o variabilă `m_lcFisiere` de categorie **Control** și tip **CListCtrl**.
- o casetă de editare, cu identificatorul-ul `IDC_CALE` (în dreapta casetei `IDC_DISC`). Toate casetele de editare au selectată opțiunea **Right aligned text** de la **Extended Styles**;
- o casetă de editare (**Total Directoare**) cu identificatorulul `IDC_TOTAL_DIR`, având asociată variabila de categorie **Value** și tip `int`, `m_nTotalDirectoare`;
- o casetă de editare (**Total Fisiere**) cu identificatorul `IDC_TOTAL_FIS`, având asociată variabila de categorie **Value** și tip `int`, `m_nTotalFisiere`;
- o casetă de editare (**Dimensiune Totala**) cu identificatorulul `IDC_TOTAL_DIM`, având asociată variabila de categorie **Value** și tip `long`, `m_lnTotalDimensiune`;

Similar exemplului din proiectul *wiz3*, funcția `OnInitDialog()` va fi modificată, pentru a putea popula caseta cu listă pentru afișarea directoarelor, precum și pentru stabilirea coloanelor casetei control listă.

```

BOOL CExplorerDlg::OnInitDialog()
{
    ...
    // TODO: Add extra initialization here
    PopulezCombo();
    m_lcFisiere.InsertColumn(0," Nume ", LVCFMT_LEFT, 120);
    m_lcFisiere.InsertColumn(1," Dimensiune ",LVCFMT_LEFT, 100);
    m_lcFisiere.InsertColumn(2," Data creerii ",LVCFMT_LEFT, 180);
    return TRUE; // return TRUE unless ...
}

```

Funcția de populare a casetei combinate va trebui să determine ce unități logice sunt prezente în sistem, indiferent dacă acestea sunt locale sau la distanță (în rețea). Unitățile logice odată găsite, vor trebui inserate în caseta combinată Această funcție va avea implementarea:

```

void CExplorerDlg::PopulezCombo()
{
    CString bufDisc,DiscCurent;
    for (int disc=0; disc<25;disc++)
    {
        bufDisc.Format("%c:", disc+'A');
        if ((GetDriveType(bufDisc+"\\")==DRIVE_REMOVABLE) ||
            (GetDriveType(bufDisc+"\\")==DRIVE_FIXED) ||
            (GetDriveType(bufDisc+"\\")==DRIVE_CDROM) ||
            (GetDriveType(bufDisc+"\\")==DRIVE_REMOTE))
            m_cbDisc.AddString(bufDisc);
    }
    DiscCurent.Format("%c",_getdrive()+'A'-1);
    int nIndex=m_cbDisc.SelectString(-1,DiscCurent);
    if (nIndex==CB_ERR)

```

```

{
    CString ErrStr=" Disc "+DiscCurent+
        " nepregatit sau violare acces";
    MessageBox(ErrStr, "EROARE",MB_ICONSTOP);
}
OnSelchangeDisc();
}

```

Am folosit funcțiile:

- **UINT GetDriveType(LPCTSTR lpRootPathName)** – determină tipul unității logice curente. *lpRootPathName* este un șir de caractere specificând directorul rădăcină al unității logice despre care se culeg informații. Funcția returnează tipul supărului unității logice. Valorile care ne interesează sunt:
 - **DRIVE_REMOVABLE** – unitatea logică este asociată unui disc ce poate fi scos din unitatea locală de disc (floppy);
 - **DRIVE_FIXED** – unitatea logică este asociată unui disc de tip hard-disc local;
 - **DRIVE_CDROM** – unitatea logică este asociată unui CD-ROM;
 - **DRIVE_REMOTE** – unitatea logică este asociată unei mapări de rețea;;
- **int _getdrive()** – returnează valoarea unității de disc curent selectate (**A**=1, **B**=2, etc.). Această funcție se găsește în biblioteca *<direct.h>*, pe care va trebui să o inserăm în fișierul sursă;

Schimbarea selecției discului, trebuie să aibă ca efect afișarea în caseta cu listă a subdirectoarelor din directorul rădăcină a discului respectiv. Vom asocia deci mesajului **CBN_SELCHANGE** generat de caseta combinată funcția (cu **ClassWizard**) **OnSelchangeDisc()**. Vom adăuga de asemenea clasei **CExplorerDlg** variabilele membru **CString m_strDisc**, care va conține permanent calea spre directorul curent, **POSITION pos**, **CStringList m_strLista** și **BOOL m_bInainte**. În lista de șiruri stocată în **m_strLista** vom salva calea spre fiecare director selectat, astfel încât la revenire în directorul părinte, să regăsim vechea cale din listă. **m_bInainte** va fi **TRUE** dacă intrăm într-un subdirector și respectiv **FALSE** dacă revenim în directorul părinte.

```

void CExplorerDlg::OnSelchangeDisc()
{
    // TODO: Add your control notification handler code here
    int nIndex=m_cbDisc.GetCurSel();
    if (nIndex != CB_ERR)
    {
        m_cbDisc.GetLBText(nIndex,m_strDisc);
        m_strLista.RemoveAll();
        pos=m_strLista.GetHeadPosition();
        m_bInainte=TRUE;
        PopulezDirector();
    }
    else MessageBox(" Eroare selectie disc ", "EROARE",MB_ICONSTOP);
}

```

Când selectăm un nou subdirector, va trebui să populăm și caseta cu listă cu subdirectoarele acestuia. Deci, în funcția asociată acestui eveniment va trebui să apelăm funcția **PopulezDirector()**:

```

void CExplorerDlg::PopulezDirector()
{

```

```

m_lbDirector.ResetContent();
m_nTotalDirectoare=0;
HANDLE hGasesc;
WIN32_FIND_DATA dataGasesc;
BOOL bMaiSunt=TRUE;
GetDlgItem(IDC_CALE)->SetWindowText(m_strDisc);
if (m_bInainte) m_strLista.AddTail(m_strDisc);
hGasesc=FindFirstFile(m_strDisc+"\\*.*",&dataGasesc);
if (hGasesc==INVALID_HANDLE_VALUE)
{
    CString ErrStr=" Disc "+m_strDisc+
        " nepregatit sau violare acces";
    MessageBox(ErrStr, "EROARE",MB_ICONSTOP);
}
else
{
    while (bMaiSunt)
    {
        if (dataGasesc.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
        if (strcmp(dataGasesc.cFileName, "."))
        {
            if ((m_strLista.GetCount()>1)
                || (strcmp(dataGasesc.cFileName, "..")))
                m_lbDirector.AddString(dataGasesc.cFileName);
            if (strcmp(dataGasesc.cFileName, ".."))
                m_nTotalDirectoare++;
        }
        bMaiSunt=FindNextFile(hGasesc, &dataGasesc);
    }
    FindClose(hGasesc);
    PopulezFisiere();
}
}

```

Funcția rezolvă mai multe probleme:

- afișează directorul curent în caseta de editare IDC_CALE;
- dacă funcția de căutare a primului fișier din directorul rădăcină al unității logice selectate returnează un cod de eroare, să afișeze mesajul de eroare;
- afișează doar acele nume de fișiere care au atributul de director, mai puțin directorul “.”;
- o problemă apare în cazul mapărilor de rețea. În acest caz, un director apare ca o unitate logică, deci va apare ca o literă în caseta combinată IDC_DISC. Dacă acest director nu este directorul rădăcină, el va conține și numele “.” de revenire în directorul părinte. Dar, pentru o unitate logică, acesta nu există. Un dublu click pe revenire în directorul părinte va avea în acest caz ca efect producerea unei erori de aserție, deci funcționarea anormală a programului. Pentru a se evita aceasta, în cazul mapărilor de rețea se evită afișarea numelui “.”. Situația este sesizată prin faptul că numele detectat de FindFirstFile() sau FindNextFile() este “.”, dar lista de cale are un singur element (deci suntem în rădăcină);
- trebuie să apeleze funcția de afișare a fișierelor din director, PopulezFisiere(), care va trebui adăugată clasei CExplorerDlg ca și funcție membru:

```

void CExplorerDlg::PopulezFisiere()
{
    m_lcFisiere.DeleteAllItems();
    m_nTotalFisiere=0;
}

```

```

m_lnTotalDimensiune=0;
HANDLE hGasesc;
WIN32_FIND_DATA dataGasesc;
BOOL bMaiSunt=TRUE;
int nItem=0;
long int nDimFis;
CTime nCreTime;
CString strBuffer;
hGasesc=FindFirstFile(m_strDisc+"\\*.*", &dataGasesc);
while (bMaiSunt)
{
    if (!(dataGasesc.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY))
    {
        nDimFis=0;
        nItem=m_lcFisiere.InsertItem(0, dataGasesc.cFileName);
        nDimFis += (dataGasesc.nFileSizeHigh*MAXWORD)
            + dataGasesc.nFileSizeLow;
        m_lnTotalDimensiune += nDimFis;
        strBuffer.Format("%d",nDimFis);
        m_lcFisiere.SetItemText(nItem, 1, strBuffer);
        nCreTime=CTime(dataGasesc.ftLastWriteTime, 0);
        strBuffer = nCreTime.Format("%A, %B %d, %Y");
        m_lcFisiere.SetItemText(nItem, 2, strBuffer);
        m_nTotalFisiere++;
    }
    bMaiSunt=FindNextFile(hGasesc, &dataGasesc);
}
FindClose(hGasesc);
UpdateData(FALSE);
}

```

Pentru afișarea datei fișierelor, se utilizează un obiect de clasă `ctime`, despre care vom vorbi pe larg în capitolul următor

La un dublu click asupra unui director trebuie să intrăm directorul respectiv. Deci casetei cu listă ce afișează directoarele, va trebui să îi asociem funcția `OnDblclkDirector()`, activată de mesajul `LBN_DBLCLK`.

```

void CExplorerDlg::OnDblclkDirector()
{
    // TODO: Add your control notification handler code here
    CString strTemp;
    int nIndex=m_lbDirector.GetCurSel();
    if (nIndex != CB_ERR)
    {
        m_lbDirector.GetText(nIndex, strTemp);
        if (strcmp(strTemp,"..")) {
            m_bInainte=TRUE;
            m_strDisc += "\\\" + strTemp;
        }
        else
        {
            m_bInainte=FALSE;
            pos=m_strLista.GetTailPosition();
            m_strLista.GetPrev(pos);
            m_strDisc=m_strLista.GetAt(pos);
            m_strLista.RemoveTail();
        }
    }
    else MessageBox(" Eroare selectie drive ", "EROARE", MB_ICONSTOP);
    PopulezDirector();
}

```

```
}

```

Dacă apăsmă dublu click pe “..”, funcția setează `m_bInainte=FALSE`, selectează calea penultimă din listă și elimină coada listei. Altfel, face `m_bInainte= TRUE` și adaugă căii numele noului director.

REZUMAT

Tabelul 6.4

Variabile adăugate cu ClassWizard		Funcții adăugate cu ClassWizard
CcomboBox	<code>m_cbDisc;</code>	<code>OnSelchangeDisc()</code>
CListBox	<code>m_lbDirector</code>	<code>OnDblclkDirector()</code>
CListCtrl	<code>m_lcFisiere</code>	
long	<code>m_lnTotalDimensiune</code>	
int	<code>m_nTotalDirectoare</code>	
int	<code>m_nTotalFisiere</code>	
Variabile membre <i>CExplorerDlg</i>		Funcții membre <i>CExplorerDlg</i>
CString	<code>m_strDisc</code>	<code>PopulezCombo()</code>
CStringList	<code>m_strLista</code>	<code>PopulezDirector()</code>
POSITION	<code>pos</code>	<code>PopulezFisiere()</code>
BOOL	<code>m_bInainte</code>	

Întrebări și probleme propuse

1. Implementați și executați toate exemplele propuse în capitolul 6;
2. Modificați programul astfel încât să puteți tasta în caseta combinată o cale de directoare și aceasta să fie preluată și interpretată de program;
3. Afișați în controlul de tip listă și informații cu privire la data creerii fișierelor;
4. Inerați un nou control de tip listă, în care, la selecția unui subdirector din arbore, să afișați numărul total de subdirectoare din director, numărul total de fișiere și dimensiunea totală a acestora;