

Laborator VIII

APLICAȚII WINDOWS – Controale de tip Buton și Controale de Editare

1. Scopul lucrării

Lucrarea de laborator are ca scop familiarizarea cu modul de creare a unei aplicații Windows utilizând biblioteca de clase MFC în mediul Microsoft Visual C++ 2010.

2. Controale de tip buton

Un buton este un tip particular de fereastră ce conține un text sau o etichetă de tip imagine, folosită în general într-o casetă de dialog, într-o bară de instrumente sau în alte tipuri de ferestre ce conțin controale. În general butoanele se utilizează pentru a indica o alegere făcută de utilizator.

Există cinci tipuri de butoane furnizate de Windows:

- Butoane de comandă (*Buttons*) – ce au un aspect tridimensional, creând impresia că sunt apăstate atunci când se face click cu mouse-ul. Uzual au atașat un text pe buton.
- Butoane radio (*Radio buttons*) – au un aspect de buton rotund, cu o etichetă alăturată. Se folosesc pentru cazurile în care se face o selecție între alternative mutual exclusive. Dacă butoanele radio sunt grupate împreună, numai unul dintre ele poate fi selectat.
- Casete de marcare (*Check boxes*) – sunt casete pătrate ce conțin un semn de marcare (bifare) atunci când sunt selectate, având o etichetă atașată. Se folosesc pe post de flag-uri booleene pentru a indica faptul că o anumită condiție este adevărată sau falsă. Spre deosebire de butoanele radio, chiar dacă sunt grupate, mai multe dintre ele pot fi bifate.
- Butoane desenate de utilizator (*Owner-drawn*) - sunt butoane particularizate, desenate de programator în locul celor puse la dispoziție de Windows.
- Casete grup (*Group boxes*) – sunt simple dreptunghiuri folosite pentru a încadra alte controale ce au o destinație comună. Prin intermediul lor se grupează din punct de vedere logic controale folosite în scopuri similare, ajutând utilizatorul să înțeleagă relația dintre controale.

2.1. Suportul MFC pentru butoane

În mod uzual, controalele de tip butoane se crează ca parte a unei casete de dialog. După ce se adaugă un buton, se poate folosi utilitarul ClassWizard pentru crea obiecte din clasa `Cbutton`, asociate fiecărui buton în parte și pentru a adăuga funcții care să trateze evenimentele create prin acționarea butoanelor (respectiv bifarea sau selectarea).

2.2. Exemplu de proiect ce utilizează butoane

Pentru a ilustra folosirea controalelor de tip buton se va crea un proiect bazat pe caseta de dialog, cu numele `Button`.

Se deschide editorul de casete dialog (se crează un proiect MFC Application, de tip Dialog Based). Se observă apariția barei de instrumente de control (*Toolbox*), ce conține butoanele și casetele necesare, așa cum se prezintă în Figura 1. În paleta de controale există patru tipuri diferite de pictograme pentru butoane.

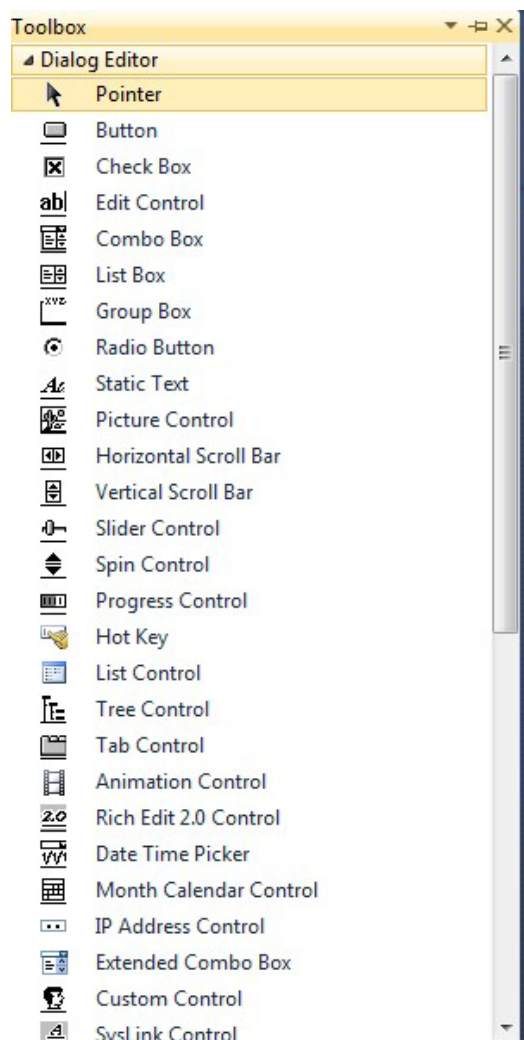


Figura 1. Caseta de instrumente de control

Pentru a adăuga unei casete un control de tip buton se parcurg următorii pași:

- se “trage” controlul dorit din Toolbox și se plasează cu mouse-ul în poziția dorită (drag-and-drop).
- Se selectează controlul dorit cu un click în paleta de controale. Apoi se face click în locația dorită din caseta de dialog.

După plasarea controlului se poate folosi mouse-ul pentru a îl re poziționa și eventual redimensiona .

Pentru a ilustra procedeul adăugați într-o casetă de dialog (`IDD_BUTTON_DIALOG`) cinci butoane, în conformitate cu dispunerea sugerată în Figura 2.

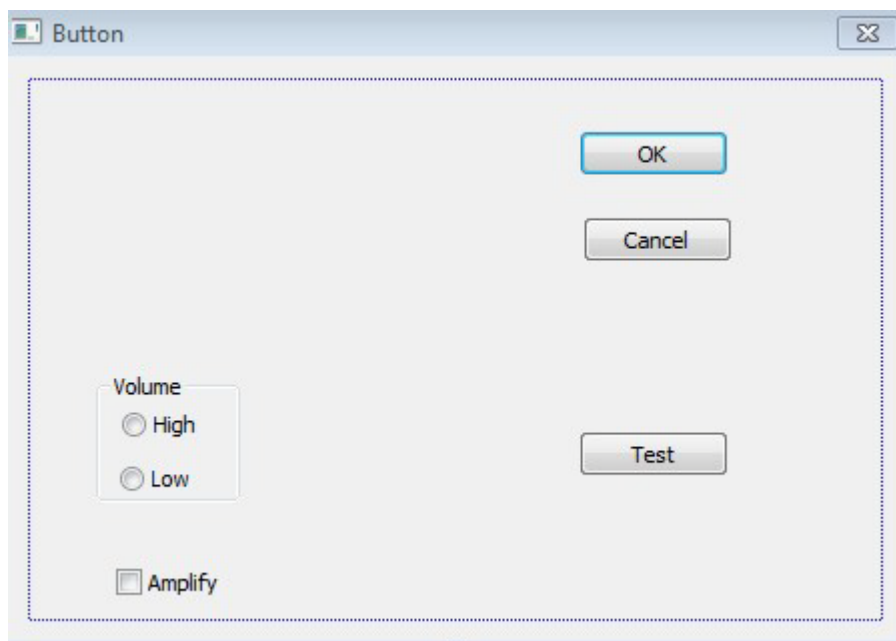


Figura 2. Caseta de dialog principală utilizată în proiectul Button

Folosiți valorile din Tabelul 1 pentru a seta proprietățile fiecărui control. Exceptând rubrica ID și titlul casetei, toate controalele folosesc setul implicit de proprietăți.

ID-ul controlului	Tipul butonului	Titlu
IDC_BTN_TEST	Button	&Test
IDC_RADIO_HIGH	Radio-button	&High
IDC_RADIO_LOW	Radio-button	&Low
IDC_GROUP_VOLUME	Group-box	&Volume
IDC_CHECK_AMP	Check- box	&Amplified

Tabelul 1. Valorile utilizate pentru controalele din IDD_BUTTON_DIALOG

2.3. Proprietățile controalelor de tip buton

Proprietățile oricărui control definesc modul său de comportare și pot fi afișate selectând **Properties** din meniul ce apare la acționarea butonului din dreapta al mouse-ului în dreptul controlului.

Proprietățile comune controalelor sunt:

- **Identitate (ID):** folosit pentru identificarea resursei. Mediul Visual Studio furnizează un ID implicit, de exemplu IDC_BUTTON1. Folosirea prefixului IDC_ pentru ID-ul unei resurse de tip control este o convenție a Microsoft.
- **Titlu (Caption):** Indică textul care apare ca etichetă a butonului (implicit Button). Pentru a face una din literele titlului o cheie mnemonică, ea trebuie precedată de ampersand (&).
- **Vizibil (Visible):** Indică faptul că butonul este inițial vizibil.
- **Dezactivat (Disabled):** Indică faptul că butonul trebuie inițial dezactivat.
- **Grup (Group):** Marchează primul control într-un grup. Toate controalele care urmează unui control cu un astfel de atribut sunt considerate ca făcând parte din același grup dacă acest check

box este nemarcat. Utilizatorul se poate deplasa între controalele din același grup cu ajutorul tastelor cu săgeți.

- Tab Stop: Indică faptul că respectivul control poate fi accesat prin acționarea tastei Tab.
- Buton Implicit (Default Button): Marchează controlul ca fiind butonul implicit. Poate exista doar un singur astfel de buton și se consideră că el este acționat atunci când utilizatorul acționează tasta Enter fără a utiliza alte controale din caseta de dialog.
- Desenat de utilizator (Owner Draw): Indică faptul că butonul va fi desenat de utilizator.

Casetele Grup(Group boxes) posedă cele mai puține proprietăți (toate exceptând `Default Button` și `Owner Draw`).

Butoanele Radio nu folosesc proprietatea de Buton Implicit, în schimb au două proprietăți ce nu sunt disponibile butoanelor de tip Button:

- Auto: Schimbă automat starea controlului atunci când este selectat . Automatically changes the state of the control when it is selected.
- Text la stânga (Left Text): Plasează eticheta controlului la stânga.

Check boxes au aceleași proprietăți ca și Butoanele radio, plus un atribut suplimentar:

- Trei stări (Tri-state): Check box-ul poate avea trei stări în loc de două. În plus, față de `true` și `false`, controlul poate fi dezactivat, cu semnificația că nu e nici `true` nici `false`.

Există câteva butoane care sunt folosite frecvent în casetele de dialog ce conțin controale. Deoarece semnificația lor este standardizată este recomandabilă respectarea acestora:

- OK: Folosit pentru a închide și accepta orice informație prezentă în caseta de dialog.
- Cancel: Folosit pentru a închide caseta de dialog și a înlătura orice modificări făcute cât timp caseta de dialog a fost deschisă. Dacă există modificări ce nu pot fi anulate, se recomandă schimbarea etichetei butonului în Close.
- Close: folosit pentru a închide caseta de dialog. Nu implică în mod necesar o acțiune din partea programului. Close se utilizează atunci când butonul Cancel nu poate fi folosit pentru a anula modificările făcute atâta timp cât caseta a fost deschisă.
- Help: Folosit pentru a cere ajutor(help) contextual.

Apply: Folosit pentru a realiza modificări bazate pe datele introduse în caseta de dialog. Spre deosebire de butonul OK, caseta de dialog trebuie să rămână deschisă și după acționarea butonului Apply.

2.4. Legarea unui Control de tip Buton de un obiect de tip Cbutton

Cel mai simplu mod de a extrage valoarea unui control este de a-l asocia cu o dată membru a unei clase folosind ClassWizard. Atunci când se asociază o dată membru unui control există două variante posibile : asocierea cu controlul însuși sau cu valoarea controlului, datele membru reprezentând butoane sunt rareori asociate cu valori. Cel mai frecvent se utilizează clasa `CButton` pentru a reprezenta cele mai multe controale de tip buton. Asocierea cu valoare se utilizează în controale de tip Edit.

Pentru a adăuga o dată membru unei clase derivate din `CDialog`, se parcurg următorii pași:

1. Se deschide ClassWizard.
2. Se selectează Member Variables.
3. Se selectează clasa derivată din `CDialog` care controlează caseta de dialog, în acest caz, `CButtonDlg`. (vezi Figura 3)
4. Se selectează ID-ul controlului ce reprezintă controlul asociat cu noua dată membru.
5. Se acționează butonul Add Variable. Apare caseta de dialog Add Member Variable.

6. Se introduce numele controlului, categoria, tipul variabilei și se acționează OK.
7. Se închide ClassWizard.

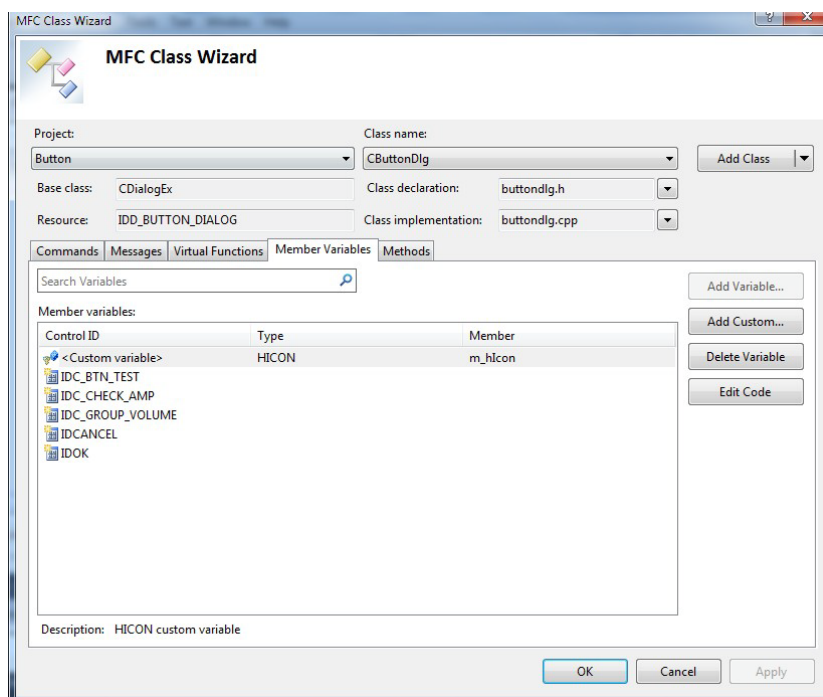


Figura 3. Fereastra Class Wizard

Urmați acești pași pentru controalele din `IDD_BUTTON_DIALOG`, folosind valorile din Tabelul 2 pentru fiecare dată membru adăugată clasei `CButtonDlg`.

ID-ul controlului	Numele variabilei	Categorie	Tip
IDC_BTN_TEST	m_btnTest	Control	CButton
IDC_GROUP_VOLUME	m_btnVolume	Control	CButton
IDC_CHECK_AMP	m_btnAmp	Control	CButton

Tabelul 2 Valori utilizate pentru a adăuga date membru pentru `CButtonDlg`

ClassWizard adaugă automat datele membru în declarația de clasă pentru `CButtonDlg`.

2.5. Asocierea de evenimente pentru butoanele unei casete de dialog

Butoanele de comandă (Buttons) sunt în mod uzual asociate cu evenimente în cadrul clasei aferente casetei de dialog. Pentru a adăuga un eveniment asociat unui buton, de exemplu, `IDC_BTN_TEST` se parcurg următorii pași:

1. Se dă dublu click pe butonul Test.
2. Se editează corpul funcției `CButtonDlg::OnBnClickedBtnTest()` în conformitate cu Listingul 1.1 și apoi compilați și executați proiectul.

Listingul 1.1.. Funcția membru `CButtonDlg::OnBtnTest`.

```
void CButtonDlg::OnBnClickedBtnTest()
{
```

```
// TODO: Add your control notification handler code here
AfxMessageBox(_T("Test button pressed"));
}
```

Observație : Butoanele Radio (*radio buttons*) și Casetele de marcarea (*Check boxes*) utilizează mai puțin mesajele BN_CLICKED messages.

2.6. Schimbarea etichetei unui buton dialog

Reamintim faptul ca, asemeni oricărui control, un buton este un tip special de fereastră. Biblioteca MFC folosește clasa `CWnd` drept clasă de bază pentru toate clasele asociate controalelor. Pentru a schimba eticheta unui buton se folosește funcția `SetWindowText`. Această funcție se folosește pentru a schimba eticheta unui buton după ce caseta de dialog a fost creată.

Se poate folosi funcția `SetWindowText` pentru a schimba eticheta butonului Amplify din exemplul studiat în Record. Pentru aceasta, se înlocuiește funcția `CButtonDlg::OnBnClickedBtnTest()` cu cea din listingul 1.2.

Listingul 1.2. Schimbarea etichetei unui buton

```
void CButtonDlg::OnBnClickedBtnTest()
{
    // TODO: Add your control notification handler code here
    static BOOL bSetWaterLevel = TRUE;
    if( bSetWaterLevel == TRUE )
    {
        m_btnVolume.SetWindowText(_T( "&Water Level" ));
        m_btnAmp.SetWindowText(_T( "&Record" ));
        bSetWaterLevel = FALSE;
    }
    else
    {
        m_btnVolume.SetWindowText(_T("&Volume" ));
        m_btnAmp.SetWindowText(_T( "&Amplify" ));
        bSetWaterLevel = TRUE;
    }
}
```

Se va observa că în grupul de butoane radio apare o alternanță de denumiri între Volume și Water Level.

2.7. Activarea și dezactivarea butoanelor

Majoritatea controalelor sunt activate (*enabled*) în mod implicit (deși există și posibilitatea setării atributului de dezactivare (*disabled*) din lista de proprietăți Clasa `CWnd` include funcția membru `EnableWindow` care permite unui obiect de tip `CWnd` să fie activat sau dezactivat. Având în vedere că toate clasele ce implementează controale, inclusiv `CButton` sunt derivate din `CWnd`, rezultă că ele beneficiază de toate datele și funcțiile membru ale clasei de bază (`CWnd`) și deci, pentru dezactivarea unui buton, se poate scrie:

```
pButton->EnableWindow( FALSE ); // Disables control
```

Parametrul pentru `EnableWindow` este `TRUE` dacă se dorește activarea controlului și `FALSE` pentru dezactivare. Parametrul implicit pentru `EnableWindow` este `TRUE`:

```
pButton->EnableWindow(); // Enables control
```

Este o practică uzuală de a activa sau dezactiva butoane sau alte controale la producerea unor evenimente recepționate de caseta de dialog. De exemplu, acționând un buton se pot activa sau dezactiva alte butoane.

În exemplul considerat, pentru a ilustra dezactivarea unui control, se înlocuiește funcția `CButtonDlg::OnBtnTest` cu codul sursă furnizat în Listingul 1.3.

Listingul 1.3. folosirea metodei `CWnd::EnableWindow` pentru a dezactiva un control dintr-o casetă de dialog.

```
void CButtonDlg::OnBnClickedBtnTest()
{
    // TODO: Add your control notification handler code here
    static BOOL bEnableControl = FALSE;

    m_btnAmp.EnableWindow( bEnableControl );

    if( bEnableControl == TRUE )
        bEnableControl = FALSE;
    else
        bEnableControl = TRUE;
}
```

La acționarea butonului Test, caseta de marcare Amplify este dezactivată. La un nou click pe Test, aceasta se activează.

2.8. Ascunderea unui buton

Operația de ascundere a unui buton localizat într-o casetă de dialog poate fi utilă în anumite aplicații (uneori, butonul are din start setată proprietatea de a fi implicit ascuns). Pentru operația de ascundere a unui buton este utilă din nou moștenirea metodelor clasei de bază, `CWnd`, mai precis folosirea metodei `CWnd::ShowWindow` pentru a ascunde butonul `pButton`, astfel:

```
pButton->ShowWindow( SW_HIDE ); // Hide control
```

Pentru a afișa o fereastră ascunsă, metoda `ShowWindow` se folosește cu parametrul `SW_SHOW`:

```
pButton->ShowWindow( SW_SHOW ); // Display control
```

Listingul 1.4 furnizează un exemplu de folosire a metodei `CWnd::ShowWindow` pentru a ascunde și respectiv afișa alternativ o serie de butoane din caseta de dialog principală.

Listingul 1.4. Folosirea `CWnd::ShowWindow` pentru a ascunde un control.

```
void CButtonDlg::OnBnClickedBtnTest()
{
    // TODO: Add your control notification handler code here
    static int nShowControl = SW_HIDE;

    m_btnAmp.ShowWindow( nShowControl );

    if( nShowControl == SW_SHOW )
        nShowControl = SW_HIDE;
    else
        nShowControl = SW_SHOW;
}
```

2.9. Definirea și prescrierea ordinii de acces cu tasta Tab

La deschiderea unei casete de dialog, unul dintre controale va fi implicit comandat de la tastatură (echivalent, va avea *keyboard focus*, sau pe scurt va avea *focus*), fapt semnalat prin apariția unui dreptunghi trasat cu linie punctată în jurul său.

Utilizatorul poate schimba focus-ul către un alt control acționând tasta Tab. De fiecare dată când se acționează tasta Tab, un control nou primește focus.

Controalele se selectează într-o ordine fixată numită *tab order*. Aceasta permite selectarea controalelor fără a folosi mouse-ul.

Într-o casetă de dialog, ordinea de acces prin tasta Tab (*tab order*) este dată implicit de ordinea în care au fost definite controalele în scriptul de resurse, cel mai recent adăugat fiind ultimul în *tab order*. Se pot folosi facilitățile resurselor incluse în mediul Visual Studio pentru a schimba această ordine.

Pentru aceasta, având caseta de dialog afișată în mediu, se selectează Tab Order din meniul Format, sau se acționează Ctrl+D. Fiecare control din caseta de dialog are atributul `tabstop` căruia i se asociază un număr, așa cum se arată în Figura 4.

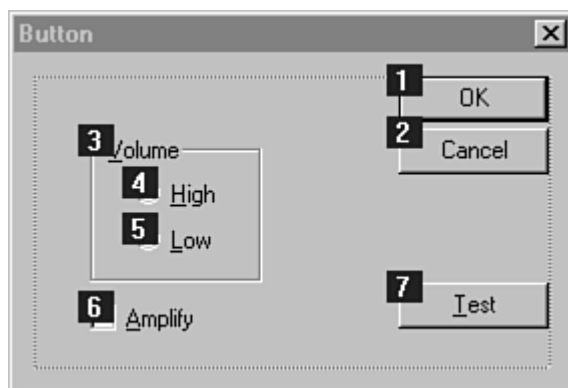


Figura 4. Afișarea ordinii de acces cu tasta Tab pentru controalele casetei de dialog

Pentru a schimba ordinea, se face click pe controlul ce trebuie să fie pe poziția 1, apoi pe celelalte, în ordinea dorită. Pe ecran apar afișate etichete corespunzătoare ordinii selectate.

3. Controale de tip editare

3.1. Controale de tip casete de editare (Edit Controls)

O casetă de editare (*edit control*) este o fereastră folosită pentru stocarea textelor introduse de utilizator, fără format impus (*free-form text input*).

Pot exista :

- casete de editare ce permit introducerea unei singure linii de text (*single-line edit control*).
- casete de editare ce permit introducerea mai multor linii de text (*multiple-line edit control*).

Pentru reducerea efortului de introducere a textelor, se pot prevedea afișări de texte implicite, precum și un minim de facilități de editare, așa cum rezultă din Tabelul 3.

Command	Keystroke
Cut	Control+X
Paste	Control+V
Copy	Control+C
Undo	Control+Z

Tabelul 3. Comenzi de editare disponibile în casete de editare

Observație : Deși casetele de editare pot genera evenimente (așa cum se întâmplă în cazul butoanelor), uzual ele se folosesc pentru a stoca date.

3.2. Suportul MFC pentru casetele de editare

Casetele de editare se adaugă unei casete de dialog într-un mod similar mecanismului prezentat în cazul controalelor de tip buton, folosind ClassWizard pentru a configura controlul. În acest scop se pot utiliza fie clasa `Cedit`, fie clasa `Cstring`.

Pentru a ilustra modul de folosire a acestui tip de controale se va utiliza un proiect de tip SDI în care se va folosi o casetă de dialog de test.

1. Se crează un proiect SDI (Single Document) cu numele EditTest folosind MFC AppWizard.

2. Se adaugă o resursă de tip casetă de dialog cu ID-ul `IDD_TEST` și titlul (caption) Test Dialog.

Folosind ClassWizard, se crează clasa `CTestDlg` pentru noua resursă de tip casetă de dialog.

3. Se adaugă un articol de meniu în meniul View cu ID-ul `ID_VIEW_TEST` și titlul Test. Se adaugă o funcție de tratare a mesajelor pentru noul articol de meniu. Se utilizează codul sursă din Listingul 2.1 pentru funcția ce tratează mesajul `CMainFrame`.

4. Se include declarația de clasă pentru `CTestDlg` în fișierul `MainFrm.cpp`:

```
#include "testdlg.h"
```

5. Se adaugă în caseta de dialog un control de tip buton, `IDC_TEST`, etichetat Test. Folosind ClassWizard, se adaugă o funcție pentru tratarea mesajului `BN_CLICKED` mesaj ce se va folosi ulterior.

6. Compilați și testați proiectul.

Listing 2.1. Tratarea selecției de meniu .

```
void CMainFrame::OnViewTest()
{
    CTestDlg    dlg;

    dlg.DoModal();
}
```

3.3. Adăugarea unui control de tip casetă de editare într-o casetă de dialog

Pentru adăugarea unui control de tip casetă de editare într-o casetă de dialog se procedează similar cazului adăugării unui buton, folosind una din cel două metode de mai jos (se va folosi ID-ul `IDC_EDIT_TEST`):

- se “trage” controlul de tip edit din Toolbox și se plasează în poziția dorită (drag-and-drop).
- se selectează controlul de tip Edit cu un click pe pictograma Edit Control din Toolbox și apoi se face click pe locația dorită.

Caseta de dialog trebuie să arate ca în Figura 5.

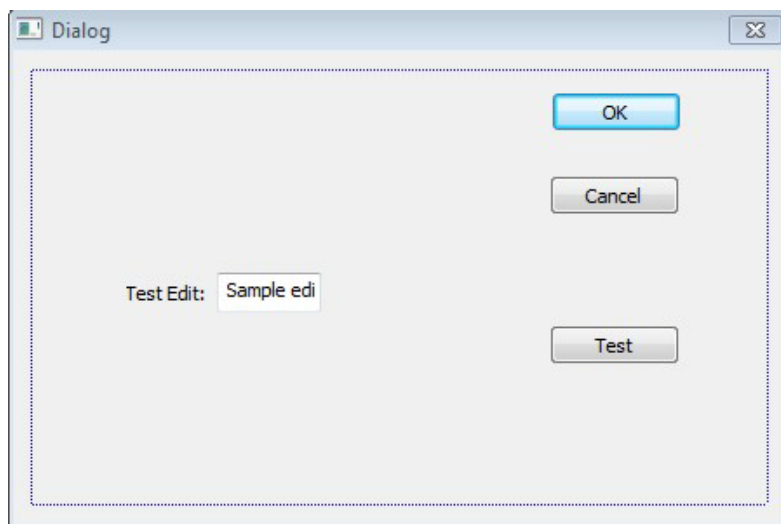


Figura 5. Caseta de dialog

Pentru controalele de tip Edit, se regăsesc proprietățile generale ale resurselor:

- ID, Visible, Disabled, Group, Help ID, Tab Stop.

dar și proprietăți specifice (listate în anexă) .

3.4. Legarea unui obiect de tip CEdit de un control de tip Edit

Așa cum s-a arătat, pentru a interacționa cu un control, se poate atașa acestuia un obiect de tip CEdit. În acest scop se utilizează ClassWizard astfel :

1. Se deschide ClassWizard.
2. Se selectează clasa derivată din Cdialog care gestionează caseta de dialog (în acest caz, CtestDlg).
3. Se selectează Member Variables.
4. Se selectează ID-ul ce reprezintă controlul asociat noii date membru (în acest caz, IDC_EDIT_TEST).
5. Se acționează butonul Add Variable și apare caseta de dialog Add Member Variable. Se introduc numele, categoria și tipul variabilei, în conformitate cu valorile din Tabelul 4.

ID-ul controlului	Numele variabilei	Categorie	Tip
IDC_EDIT_TEST	m_editTest	Control	CEdit

Tabelul 4. Valorile utilizate pentru adăugarea unei date membru de tip Cedit în CTestDlg.

3.5. Preluarea textelor dintr-un control de tip Edit folosind funcțiile membru din clasa Cedit

Pentru a inițializa un mesaj la afișarea casetei se poate folosi funcția SetWindowText apelată corespunzător în metoda CTestDlg::OnInitDialog:

```
m_editTest.SetWindowText( "Default" );
```

Pentru a ilustra modul de preluare a textelor dintr-un control de tip Edit se vor folosi două din funcțiile membru ale clasei Cedit, GetWindowText and LineLength, așa cum se arată în Listingul 2.2.

Listingul 2.2. Preluarea textelor dintr-o fereastră de editare folosind clasa CEdit.

```

void CTestDlg::OnTest()
{
    CString szEdit;
    CString szResult;

    int nLength = m_editTest.LineLength();
    m_editTest.GetWindowText( szEdit );
    szResult.Format( "%s has %d chars", szEdit, nLength );

    AfxMessageBox( szResult );
}

```

La acționarea butonului Test, textul introdus în fereastra de editare este preluat folosind obiectul `m_editTest`. În mod normal, textul introdus este de interes numai dacă se acționează OK . Dacă se acționează Cancel, caseta de dialog trebuie închisă și, uzual, informația introdusă este ignorată.

3.6. Transferul parametrilor în/din casetele de dialog folosind rutinele DDV și DDX

Rutinele DDV și DDX sunt funcții care ajută la manevrarea datelor în casetele de dialog. DDV (Dialog Data Validation) se folosește pentru validarea datelor (de exemplu pe baza lungimii șirului introdus sau a apartenenței la un anumit interval), iar DDX (Dialog Data Exchange), pentru schimbul de date înspre/dinspre controalele unei ferestre de dialog. Deși aceste rutine se pot utiliza direct, folosirea utilitarului ClassWizard este recomandabilă.

Funcțiile DDX leagă date membru din clasa asociată ferestrei de dialog cu controale conținute de fereastră, permițând datelor să fie transferate spre/dinspre control.

De exemplu, o fereastră de dialog se folosește astfel:

```

CMyDialog    dlgMine;
dlgMine.DoModal();

```

În acest exemplu, fereastra de dialog este creată la apelul funcției `DoModal`, iar funcția nu se încheie până când utilizatorul nu închide caseta de dialog. Acest aspect poate crea probleme dacă este necesar un transfer de date. Deoarece nici un control nu există până în momentul în care se crează fereastra de dialog, folosirea funcțiilor `SetWindowText`, `GetWindowText` sau a altor funcții pentru a interacționa direct cu controalele conținute în fereastră nu este posibilă (apelul lor după închiderea ferestrei de dialog este inutil pentru colectarea datelor de la utilizator).

Atunci când se utilizează o rutină DDX pentru a schimba informații cu o fereastră de dialog se poate folosi codul de mai jos:

```

CMyDialog    dlgMine;
dlgMine.m_szTest = "Hello World";
dlgMine.DoModal();

```

Rutinele DDX permit accesul la controalele unei ferestre de dialog înainte și după crearea ferestrei de dialog.

Cel mai simplu și mai util mod de adăugare a rutinelor DDV și DDX într-o casetă de dialog este oferit de ClassWizard. Datele membru asociate cu controale prin valoare folosesc în mod automat rutinele DDV și DDX furnizate de MFC. De exemplu, data membru `CString` este frecvent asociată cu un control de tip Edit. ClassWizard adaugă cod pentru manevrarea schimbului de date și validarea acestora în două locuri:

- În constructorul casetei de dialog, pentru a inițializa data membru.

- În funcția membru a ferestrei de dialog `DoDataExchange`, ClassWizard adaugă rutinele DDV și DDX pentru fiecare dată membru asociată unui control prin valoare.

`DoDataExchange` este o funcție virtuală apelată pentru a transfera date între control și datele membru ale ferestrei de dialog. `DoDataExchange` are un singur parametru, `TRUE` sau `FALSE`, `TRUE` fiind parametru implicit. La apelul `DoDataExchange(FALSE)`, data se transferă de la data membru spre control. La apelul `DoDataExchange(TRUE)`, data se copie din control în data membru..

În Figura 6 se sugerează faptul că funcția `DoDataExchange` are un singur parametru ce controlează sensul de transfer al datelor dintre control și data membru `m_szTest`.

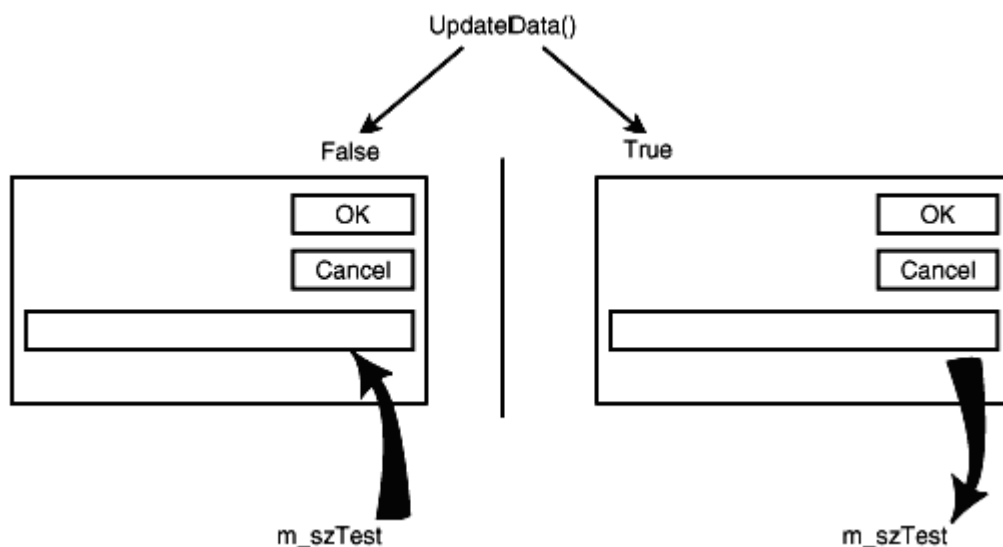


Figura 6. Rutinele DDV și DDX folosite pentru a manevra datele din casetele de dialog.

3.7. Asocierea unui control cu o dată membru prin valoare

Adăugarea unei date membru asociată prin valoare unui control se face în mod similar cazului asocierii cu categoria de tip control.

De exemplu, pentru a crea o dată membru asociată cu controlul de tip edit `IDC_EDIT_TEST` se parcurg următorii pași:

1. Se deschide ClassWizard.
2. Se selectează clasa derivată din `Cdialog` care gestionează caseta de dialog, în acest caz, `CTestDlg`.
3. Se selectează tab-ul Member Variables.
4. Se selectează ID-ul controlului ce reprezintă controlul asociat cu noua dată membru, în acest caz, `IDC_EDIT_TEST`.
5. Se acționează butonul Add Variable. Apare caseta de dialog Add Member Variable. Se introduce numele controlului, categoria și tipul variabilei, apoi OK. Pentru acest exemplu, se folosesc valorile din Tabelul 5.

ID-ul controlului	Numele variabilei	Categorie	Tip
IDC_EDIT_TEST	m_szTest	Value	CString

Tabelul 5. Valorile folosite pentru a asocia data membru CString cu un control de tip edit.

O dată membru asociată prin valoare unui control de tip edit poate fi de tip `int`, `UINT`, `long`, `DWORD`, `float`, `double`, or `BYTE` și , cel mai adesea, `CString`.

După ce se închide caseta de dialog `Add Member Variable`, `ClassWizard` afișează un control de tip edit ce poate fi folosit pentru a specifica tipul de validare dorit. (În cazul asocierii unui obiect de tip `CString`, se poate specifica lungimea maximă a șirului, iar în cazul valorilor numerice, domeniul de valori admise).

3.8. Schimbul de informații cu un control de tip Edit folosind funcțiile DDX

Datele membru asociate de `ClassWizard` controalelor din ferestrele de dialog sunt adăugate ca date membru publice, fiind în acest mod ușor de accesat și de utilizat

De exemplu, pentru a utiliza data membru `m_szTest` adăugată anterior, se editează funcția `CMainFrame::OnViewTest` în conformitate cu Listingul 2.3. (Înainte de compilare se șterge linia de mai jos, din `CTestDlg::OnInitDialog`, utilizată în exemplul anterior):

```
m_editTest.SetWindowText( "Default" );
```

Listingul 2.3. Folosirea datelor membru pentru a schimba informații cu un control de tip edit.

```
void CMainFrame::OnViewTest()
{
    CTestDlg    dlg;

    dlg.m_szTest = "DDX Test";

    if( dlg.DoModal() == IDOK )
    {
        AfxMessageBox(_T(dlg.m_szTest ));
    }
    else
    {
        AfxMessageBox(_T( "Dialog cancelled" ));
    }
}
```

Din listingul 2.3 se observă inițializarea valorii datei membru `m_szTest` înaintea afișării ferestrei de dialog. `CDialog::OnInitDialog` apelează funcția `CWnd::UpdateData`. Deoarece funcția `UpdateData` este virtuală, se apelează versiunea adecvată, și anume cea aferentă clasei derivate din `Cdialog` ce gestionează fereastra de dialog.

După închiderea ferestrei de dialog, funcția `CMainFrame::OnViewTest` verifică valoarea returnată de `DoModal`. Dacă s-a returnat `IDOK`, respectiv fereastra de dialog s-a închis prin acționarea butonului OK, se afișează valoarea datei membru `m_szTest`.

ANEXA

Proprietățile specifice controalelor de tip Edit (în engleză !).

The following properties are displayed by clicking the Styles tab in the Properties dialog box:

- **Align Text:** A drop-down list box that is enabled if the edit control is an MLE. The text can be aligned to the left, center, or right, with left as the default.
- **Multiline:** Defines the control as an MLE. This option is not selected by default.
- **Number:** Restricts the edit control to digits only. This feature is available only in Windows 95 or Windows NT version 3.51 or later.

- Horizontal Scroll: Enabled only for an MLE and provides a horizontal scrollbar. The option is not selected by default.
- Auto HScroll: Scrolls text to the right if needed. This option is normally selected.
- Vertical Scroll: Enabled only for an MLE and provides a vertical scrollbar. The option is not selected by default.
- Auto VScroll: Enabled only for an MLE and provides automatic scrolling when the user presses Return on the last line. The option is not selected by default.
- Password: Hides the user's input by displaying an asterisk instead of each character. This option is available only in single-line controls and is not selected by default.
- No Hide Selection: Changes the way an edit control handles the focus. When this option is enabled, text appears to be selected at all times. This option is not selected by default.
- OEM Convert: Performs conversions on the user's input so that the `AnsiToOem` function works correctly if called by your program. This option is not selected by default.
- Want Return: Applies to MLE controls. This option allows an edit control to accept an Enter keypress, so that an Enter keypress doesn't affect the dialog box's default pushbutton.
- Border: Creates a border around the control. This option is selected by default.
- Uppercase: Converts all input to uppercase characters. This option is not selected by default.
- Lowercase: Converts all input to lowercase characters. This option is not selected by default.
- Read-only: Prevents the user from typing or editing text in the edit control. This option is not selected by default.