Cuprins C 4 - VB

Tratarea erorilor în execuție în VB

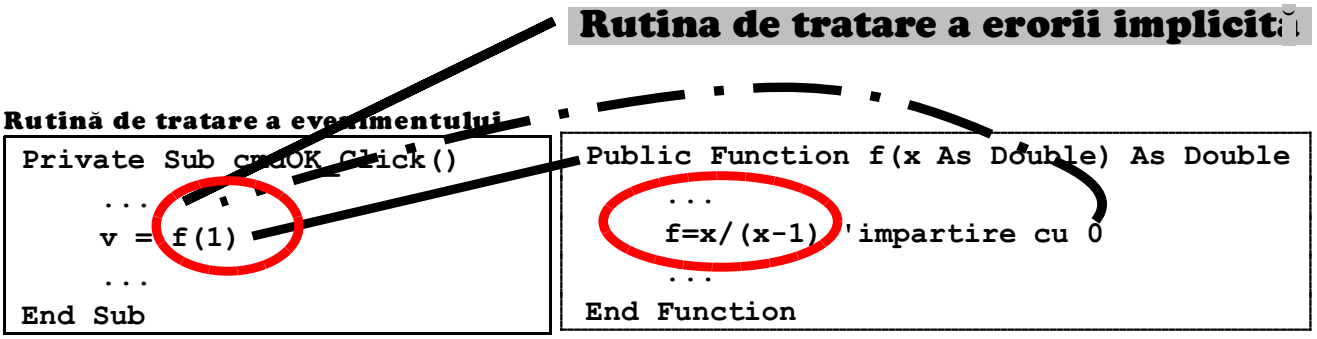
- **Tratatrea erorilor în VB**
- **Obiectul `Err`**
- **Strategii de tratare a erorilor**
- **Instrucțiunea `On Error GoTo`**
- **Instrucțiunea `On Error Resume Next`**
- **Generarea erorilor**

Aproape orice program poate întîlni erori în timpul funcționării (execuției) lui. Dacă aceste situații nu sînt anticipate și tratate prin secvențe de cod speciale ele conduc la terminarea bruscă a programului. Motivele erorilor sînt multe, epuizarea spațiului de pe disc, pierderea conexiunii la rețea, editarea unor înregistrări blocate dintr-o bază de date, calculul unor funcții matematice cu argumente greșite etc. Oricare ar fi motivul erorii, dacă se dorește maximizarea șanselor păstrării datelor și a satisfacției clientului față de aplicație apariția erorilor și tratarea lor trebuie luate în considerare

Pentru a ne putea atinge scopul trebuie să înțelegem modul în care VB tratează erorile în timpul funcționării aplicației, cum se poate interacționa cu eroarea apărută și cum anume se poate depăsi situația de eroare, eventual, dacă este cazul chiar cum să o generăm.

Tratarea erorilor în VB

- **Cascadează erorile înapoi pînă la găsirea unei rutine de tratare a erorii**
- **Dacă nu găsește rutină de tratare a erorilor se apelează rutina implicită de tratare a erorilor din VB**



La detectarea unei erori în cod VB, verifică prima oară dacă există o rutină de tratare a erorii la nivelul procedurii. Dacă aceasta lipsește consideră responsabilă de eroare linia de cod care a apelat procedura în care a apărut eroare. Din acest motiv, în lipsa procedurii de tratare a erorii VB caută o procedură de tratare a erorii în procedura apelantă, iar procesul de revenire pe această linie continuă pînă cînd fie se găsește o rutină de tratare a erorilor, fie se ajunge la ultima procedură apelantă care nu are nici ea această rutină. În acest caz, VB va trata eroarea automat apelînd rutina de tratare implicită de erori. Această variantă nu este prea sănătoasă după cum se va vedea imediat.

Mai sus funcția f este apelată din subrutina de tratare a evenimentului `Click`. În forma în care s-a scris codul va "crăpa" deoarece 1/(1-1) conduce la împărțire cu 0. Deoarece la nivelul lui f nu există o rutină de tratare a erorii, eroarea va fi tratată ca şi cum ar fi din vina codului apelant. Aici, din nou nu există o rutină de tratare a erorii, deci VB va apela rutina de tratare implicită a erorilor.

1. Rutina implicită de tratare a erorilor

Rutina de tratare implicită a erorilor afişează o fereastră de dialog conţinînd numărul erorii şi descrierea asociată la apariţia erorii. După aceasta aplicaţia este oprită.

Ea este OK dacă ne aflăm în faza dezvoltării aplicaţiei deoarece ne permite să depanăm (la apăsarea butonului **Debug**) aplicaţia şi să reparăm problema înainte de a continua dezvoltarea.

La nivelul aplicaţiei compilate problema devine gravă, o fereastră de dialog apare cu un sigur buton **OK.** La apăsarea lui aplicaţia este terminată fără nici o şansă de a salva datele sau de a stoca într-un jurnal eroarea apărută.

Din acest motiv este bine ca toate evenimentele să aibă o formă de tratare a posibilelor erori pentru aplanarea efectelor erorile neprevăzute din aplicaţie. În cele ce urmează se va discuta modul de a realiza astfel de aplicaţii.

Obiectul Err

- **Proprietăți esențiale**
  - **`Number` - nr. unic al erorii**
  - **`Description` - descriere, deseori inutilă, a erorii**
  - **`Source` - șir de indentificare a aplicației care a generat eroarea**
- **Metode**
  - **`Clear` - resetează obiectul Err**
  - **`Raise` - generează o eroare cu proprietățile specificate**

2. Obiectul Err

Toate metodele pentru detectarea şi tratarea erorilor se folosesc de obiectul **`Err`**. La apariția unei erori obiectul **`Err`** va avea proprietățile populate cu valorile corespunzătoare ultimei erori apărute.

Din punctul de vedere al codului singura proprietate folosită pentru testarea erorii este **`Number`**. Comparînd valoarea ei cu un grupe de valori aşteptate codul poate decide cu privire la modul de continuare al acțiunilor.

`Err.Description` este un mesaj corespunzător erorii particulare fiind deseori prea criptic pentru un utilizator obişnuit al aplicației.

`Err.Source` este un şir de caractere care face cunoscut numele aplicației şi uneori al procedurii în care a apărut eroarea. Valorile din **`Description`** şi **`Source`** pot fi înscrise în fişier de tipul jurnal pentru ca dezvoltatorii programului să poată depana aplicația mai uşor.

Motoda **`Clear`** resetează obiectul **`Err`**, adică **`Number`** devine **`0`**, iar **`Source`** şi **`Description`** şiruri vide. Se foloseşte numai în cazul tratării la nivel de linie a erorilor din apliacție.

Metoda **`Raise`** este foarte importantă, ea se foloseşte pentru a genera o eroare dacă codul consideră îndeplinirea unor condiții de eroare. Sintaxa lui **`Raise`** urmează să fie prezentată în acest curs.

Instrucțiunea On Errot GoTo

```
Private Sub Command1_Click()
    Print f(1)
End Sub

Public Function f(x As Double) As Variant
    On Error GoTo PrindeEroarea
    f = CDbl(x / (x - 1))
    Exit Function

PrindeEroarea: 'aici incepe rutina de tratare a erorilor
    If Err.Number = 11 Then 'impartire cu 0
        MsgBox "Impartire cu zero", vbCritical, "EROARE"
    Else                        'alta eroare
        MsgBox Err.Description, vbCritical, _
        "EROARE NECUNOSCUTA - COD:" & Err.Number
    End If
    f = False
    Resume Next
End Function
```

3. Instrucțiunea On Error GoTo

Aplicațiile VB se scriu cu rutine de eroare asemenea celei de mai sus. O procedură poate avea una sau mai multe rutine la sfîrşitul ei şi o linie de cod care activează o rutină de tratare particulară a erorii.

On Error GoTo

Instrucţiunea **`On Error GoTo identificator`** îi spune lui VB că în cazul apariţiei unei erori mai jos de acest punct să sară la codul marcat cu eticheta (**`identificator`** urmat de **`:`**). Mai multe **`On Error GoTo`** pot să apară într-o procedura pentru a permite scrierea mai multor rutine de tratare a erorilor specifice unor anumite porţiuni de cod. Pentru a inhiba rutinele de tratare a erorilor în vederea revenirii la rutina de tratare implicită a lui VB se scrie **`On Error GoTo 0.`**

Exit Function/Sub

Atunci cînd apare o eroare execuţia programului sare la punctul sepcificat în codul de tratare a erorii, dar dacă nu apare nici o eroare trebuie să evităm intrarea în rutina de tratare a erorii. Instrucţiunea **`Exit`** se foloseşte penru terminarea unei proceduri dacă totul merge bine.

Testarea lui Err.Number

Rutina de tratare a erorii verifică valoarea lui **`Err.Number`**. Utilizatorul trebuie să ştie ce

erori pot să apară şi modul în care pot fi corectate. În general, codul va conține **`If ... Then ... Else`** sau **`Select Case`** folosind pe **`Err.Number`** ca valoare de testat pentru a decide asupra instruncțiunii următoare de executat.

Resume Next

După ce o condiție de eroare a fost tratată, programul trebuie să-şi continue execuția. Reluarea continuării execuție aplicație se poate face prin următoarele instrucțiuni:

- **`On Error Resume Next`** - instrucțiunea face ca punctul de reluare a execuției să devină instrucțiunea care ar fi urmat în cazul unei execuții normale (instrucțiunea următoare celei care a generat eroarea). În exmplu, aceasta este **`Exit Function`**.
- **`On Error Resume`** - Instrucțiunea face ca punctul de reluare să fie chiar instrucțiunea generatoare a erorii (se încearcă execuția aceleişi instrucțiuni). Dacă eroarea nu a fost eliminatnă ea va conduce la intrarea într-un ciclu infinit (de aici se iese numai cu ajutorul unui contor al reluărilor).
- **`On Error Resume identificator`** - punctul de reluare al execuției devine instrucțiunea marcată cu eticheta având numele lui indentificator.
- **`Exit Sub`** sau **`Exit Function`** - instrucțiunile fac ca procedura în curs de execuție să fie terminată.

Instrucțiunea On Error Resume Next

```
Private Sub Command1_Click()
    Dim x As Double, f As Double
    On Error Resume Next
    x = Text1.Text
    f = x / (x - 1)
    Select Case Err.Number
        Case 0              'nu este eroare
        Case 11             'impartire cu zero
            Print "Impartire cu 0"
            Exit Sub
        Case 13             'nu este valoare numerica
            Print "Valoare introdusa " & vbCrLf & _
             "nu se poate converi intr-un numar"
            Exit Sub
        Case Else
            Print Err.Number, Err.Description
            Exit Sub
    End Select
    Err.Clear               'resetare obiect Err
    Print x, f
End Sub
```

4. Instrucţiunea On Error Resume Next

Stilul de tratare al erorilor prezentat mai sus se numeşte "la nivel de linie". Instrucţiunea **`On Error Resume Next`** face ca VB să ignore erorile şi să continue execuţia secvenţială. Este responsabilitatea programatorului să verifice obiectul **`Err`** pentru a prinde erorile.

Este, de asemenea, responsabilitatea lui să folosească metoda **`Clear`** pentru a reseta informaţiiloe din obiectrul **`Err`**, altfel se va găsie aceeaşi eroare la următoarea verificare (asta dacă nu a apărut între timp o nouă eroare).

Problemele acestei strategii sînt:

- dacă se omite verificarea lui **`Err`** se poate găsi o eroare care este consecinţa unei alte erori;
- codul aplicaţiei şi cel al tratării erorilor se intercalează făcînd dificilă înţelegerea şi întreţinerea.

Avanatjele metodei:

- dacă nu conteză apariţia erorilor conduce la un cod simplu. De exemplu un astfel de cod ar putea fi:

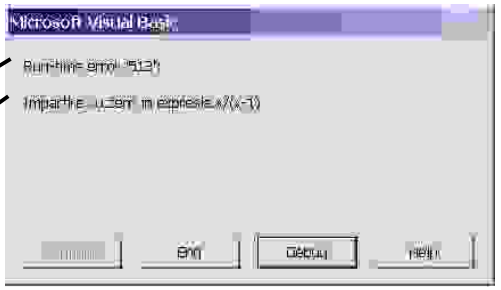
```
Dim C as Control
For Each C In Controls
     C.ForeColor = vbRed
Next
```

- este singura metodă care se poate folosi în **VBScript**.

Generarea erorilor

```
Private Sub Command1_Click()
    Dim x As Double
    Print f(Text1.Text)
End Sub

Public Function f(x As Double) As Double
    On Error GoTo rezolvaer
    f = x / (x - 1)
    Exit Function
rezolvaer:
    If Err.Number = 11 Then
        Err.Raise 513, _
            Description:="Impartire cu zero in expresia x/(x-1)", _
            Source:="Aplicatia Project1, functia f"
    Else
            Print Err.Number, Err.Description
    End If
    Resume Next
End Function
```



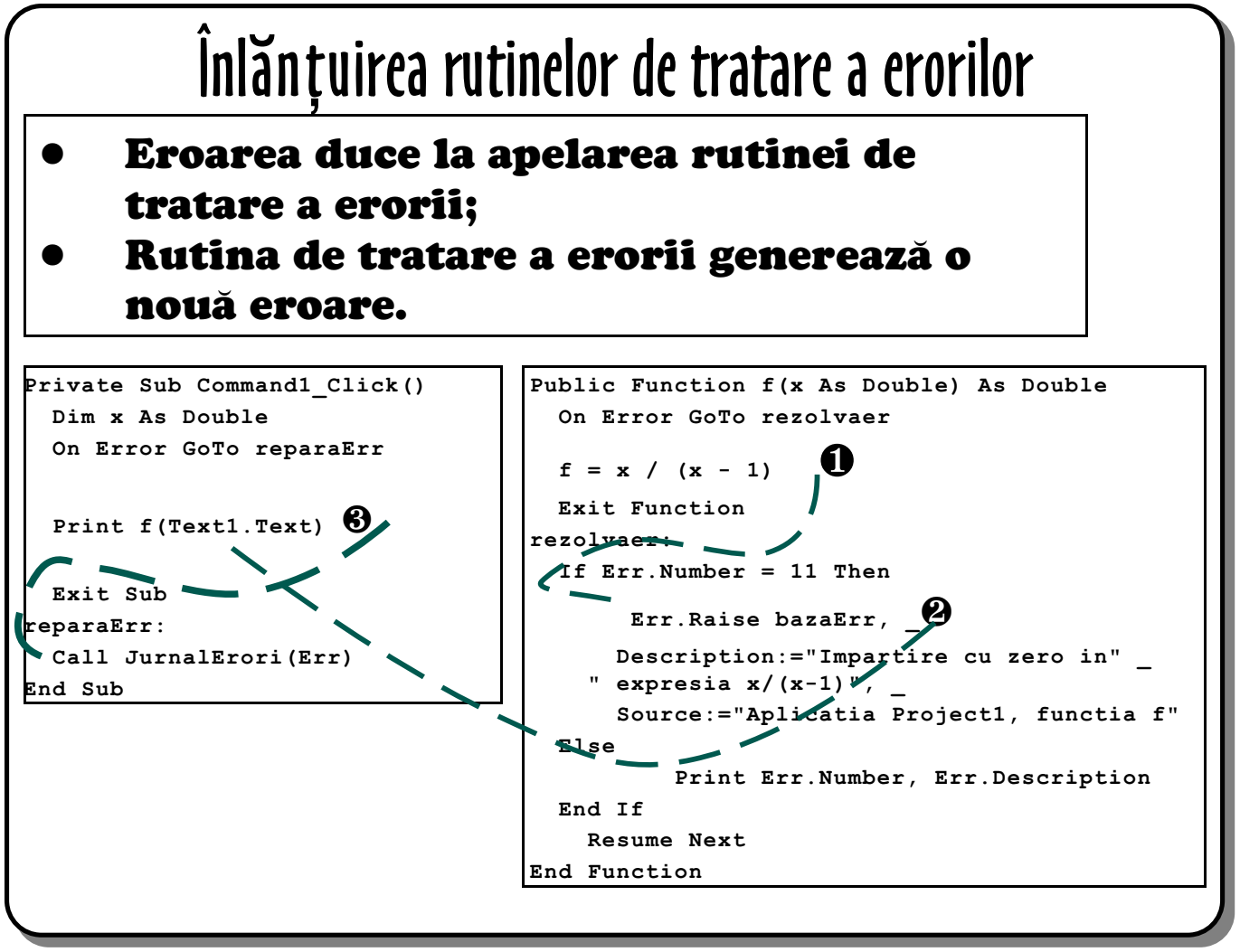
5. Generarea erorilor

VB permite generarea erorilor pentru cazul în care anumite condiţii de eroare apar. Generarea erorilor este un aspect esenţial al comunicaţiei între obiecte. Cînd o condiţie de eroare apare, trebuie oprită prelucrarea curentă (de exemplu ne intersează numărul de produse dintr-un stoc întors de o componentă, dar acesta este negativ) şi trebuie dat controlul procedurii apelante. Strategia este esenţială atunci cînd se scriu componente. La generarea unei erori trebuie creat un număr unic de indentificare al erorii şi un text care s-o descrie împreună cu textul de indentificare al sursei erorii. Acestea vor fi valori date proprietăţilor **`Number`**, **`Description`** şi **`Source`** ale obiectului **`Err`** pe care codul apelant îl primeşte.

În exeplul prezentat, s-a decis extinderea mesajului dat de rutina funcţia de calcul în cazul împărţirii cu zero. Eroarea este transferată direct subrutinei apelante deoarece apare în interiorul unei rutine de tratare a erorilor (rutinele de tratare a erorilor nu pot trata erorile generate de ele însele), care în lipsa unei rutine de tratare a erorilor pasează mai departe eroarea rutinei de tratare implicită a erorilor din VB.

Numerele de eroare

Pentru formarea unui număr de eroare valid (care nu redefineşte o eroare VB predefinită) atunci cînd se lucrează cu componente se va folosi constanta **`vbObjectError`**, la care se mai adună 512, apoi la care se mai adună valoarea pe care o dăm noi erorii. **`vbObjectError`** este ovaloare de tipul **`Long`** pe care VB o vede ca şi număr negativ(**`Const NrErr As Long =vbObjectError + 512 +1`**).



6. Înlănțuirea rutinelor de tratare a erorilor

La scrierea unei aplicații complexe trebuie să existe o strategie pentru tratarea erorilor la care ne așteptăm, la care nu ne așteptăm și la cele pe care rutina curentă de tratare a erorilor nu le poate rezolva. Următoarele reguli permit rezolvarea majorității acestor situații.

Regula 1

Nu lăsați erorile netratate într-o aplicație deoarece aceasta se va termina brusc în cazul apariției unei erori.

Regula 2

Nu folosiți formulare, ferestre de mesaje sau dialog pentru a interacționa cu utilizatorul în vederea rezolvării unei erori într-o rutină de tratare a erorilor.

Regula 3

Generați erori personalizate cu numere și descrieri utile, adică croiți erorile pe măsura aplicației.

Regula 4

Realizați o procedură de încriere a erorilor neașteptate într-un fișier text jurnal de erori sau

într-o bază de date pentru a le putea analiza mai tîrziu.

Pentru exemplul prezentat avem următoarea situație:

- Funcția **f** este apelată
- O eroare apare in calculul funcției deoarece se încearcă o împărțire cu 0 (❶)
- Rutina de tratare a erorii încearcă să trateze eroarea pe baza valorii din **Err.Number**
- Eroarea specifică are numărul 11(❷), pentru a extinde mesajul de eroare specific acestei situații se generează o nouă eroare
- Eroarea apare într-o rutină de tratare a erorilor, ea nu mai poate fi tratată de rutina curentă de tratare a erorilor şi se revine în procedura apelantă;
- Rutina de tratare a erorilor din programul apelant intră în joc (❸) pentru a rezolva situația excepțională. Parte a procesului este apelul procedurii de înregistrare a erorii într-un fişier text ce are parametru un obiect **Err**. Un exemplu de scriere al acestei proceduri ar putea fi:

```
Public Sub JurnalErori(E As ErrObject)
    Dim lnfNrFis As Long
    Dim strNumeFieEr As String
    strNumeFieEr = "C:\JurnalErori.txt"
    lnfNrFis = FreeFile()
    Open strNumeFieEr For Append Access Write As #lnfNrFis
    Write #lnfNrFis, Format$(Now, "dd/mm/yyyy - hh:mm") & _
    " Eroare: " & E.Number & ": " & E.Description & _
    " (" & E.Source & ") "
    Close #lnfNrFis
End Sub
```

iar o line din fişierul **JurnalErori.txt** din rădăcina discului **C:** este:

```
"23/02/2003 - 14:41 Eroare: -2147220991: Impartire cu zero in
expresia x/(x-1) (Aplicatia Project1, functia f) "
```

Depanarea aplicațiilor

- **Tipuri de probleme**
- **Folosirea punctelor de întrerupere (breakpoint)**
- **Instrumente pentru depanare (Debug Toolbar)**
- **Fereastra locală, imediată și de monitorizare**
- **Starea stivei (call stack)**

7. Depanarea aplicațiilor

Puține persoane sînt suficient de talentate pentru a scrie o aplicație perfectă din prima încercare. Pentru ceilalți depanarea devine un proces natural din cel al dezvoltării aplicației.

Multe tipuri de probleme pot să apară în timpul dezvoltării unei aplicații. Fiecare este unică în felul ei, dar există cîteva șabloane generale ce permit recunoașterea erorii în vederea rezolvării ei. Iată cîteva sugestii:

Eroare de aplicație
Este cazul aplicațiilor care crapă fară nici un avertisment cauzînd pierderea de date. Candidați de investigat ar fi apeluri incorecte de API, componente dezvoltate de firme terțe care conține erori. Codul problemă se identifică prin rularea pas cu pas (linie cu linie) pînă în locul în care apare eroarea. Dacă eroarea nu are explicație consultați MSDN-ul sau site-ul Microsoft pentru informații.

Programul crapă
Este cazul în care programul se termină inopinat, dar în mod repetat. Problema se rezolvă implementînd rutine de tratare a erorilor pentru fiecare eveniment care duce la terminarea incorectă a aplicației. Documentați valorile de date şi secvențele logice care conduc la eroare şi testați secvența cu probleme separat.

Comportament incorect
Dacă programul rulează, dar nu face anumite lucruri corect atunci este clar vorba despre o eroare de programare. Programatorul poartă vina şi el trebuie să o cercteze prin vizualizarea valorilor intermediare luate de variabile care ajung să fie incorecte. Eventual, vor fi prinse greşelile de logică.

Modul de rulare cu întrerupere

- **Rularea este suspendată atunci cînd:**
  - **se apasă Ctrl-Break în rulare;**
  - **codul ajunge la o linie marcată cu un punct de întrerupere;**
  - **codul ajungfe la o instrucţiune Debug.Assret validă**
- **Bascularea punctelor de întrerupere:**
  - **tasta F9;**
  - **clic pe marginea din stînga**
  - **din butonul de Toolbar.**

8. Modul de rulare cu întrerupere

Majoritatea formelor de depanare implică rularea unui program şi oprirea lui pentru a-l suspenda într-o stare numită "mod de întrerupere" (**break mode**).

Dacă aplicaţia întîlneşte o eroare netratată în timp ce este rulată în mediul VB ea intră automat în modul de întrerupere. Orice aplicaţie poate fi întreruptă prin **Ctrl-Break**, dar aceasta trebuie făcută dacă:

- aplicaţia aşteaptă un răspuns din partea utilizatorului (dacă programul este în curs de rulare oprirea lui se va face într-un punct arbitrar)
- aplicaţia s-a oprit ca răspuns la acţiunea utilizatorului deoarece codul este prins într-o buclă infinită.

Utilizatorul va seta în aceste cazuri puncte de întreruprere în program. Programul se va rula normal pînă cînd ajunge la punctul de întrerupere, moment în care aplicaţia intră în modul de întrerupere. Un cerc maro în stînga liniei de cod indică un prunct de întrerupere. O săgeată galbenă în margine pe un fond galben indică următoarea instrucţiune executabilă.

Setarea şi ştegerea punctelor de întrerupere

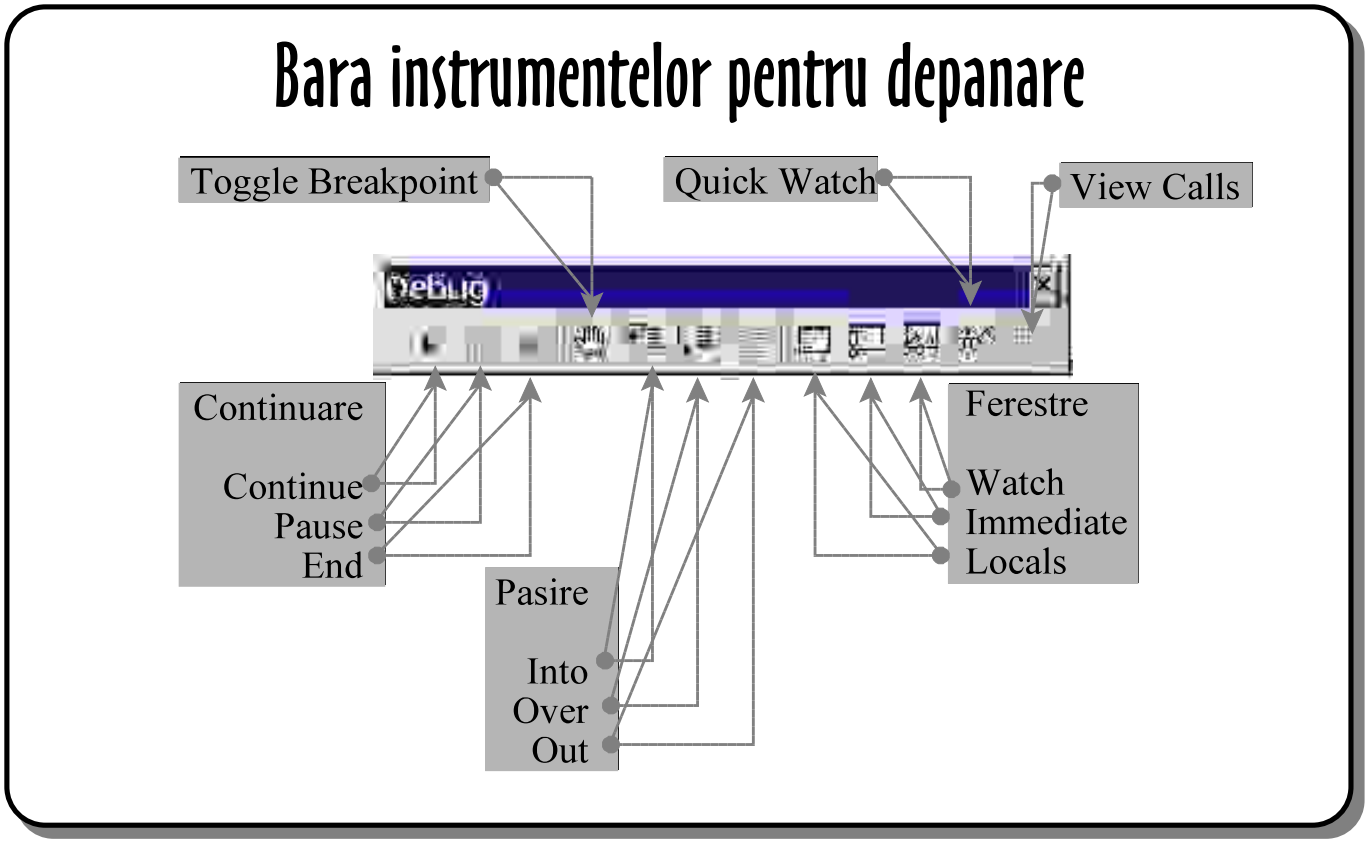
Există mai multe metode pentru setarea punctelor de întrerupere:

- apăsarea tastei **F9** va face bascularea stării de activ/inactiv a punctului de întrerupere

din linia curentă;
- clic pe marginea gri din stînga lîngă linia de cod face bascularea punctului de întrerupere;
- simbolul grafic mînă din **Debug Toolbar** face şi el bascularea punctului de întrerupere din linia curentă de cod;
- clic pe butonul drept de mouse într-o linie de cod permite inserarea sau ştergerea unui punct de întrerupere prin opțiunea *Toggle*;
- folosiți opțiunea *Toggle Breakpoint* din meniul *Debug* ...
- cuvîntul cheie `Stop` în cod are efectul unui punct de întrerupere (nu-i recomandat, trebuie scos cu mîna înainte de compilarea codului)
- instrucțiunea `Debug.Assert` poate fi folosită pentru întreruperea condiționată a codului. Este sigură deoarece este ignorată la compilare. Are ca argument o expresie `Boolean` de exemplu, `Debug.Assert i > 12`.

Modificarea următoarei instrucțiuni de executat

Cînd aplicația este în mod de întrerupere, linia galbenă este următoarea linie de cod ce va fi executată. Ea poate fi schimbată prin:
- tragerea săgeții din stînga liniei galbene pe o altă linie de cod;
- folosirea lui **Ctrl-F9** pentru a face linia curentă (pe care se află cursorul de text) următoarea instrucțiune de executat;
- folosirea lui *Set Next Statement* din meniul *Debug*...



9. Bara instrumentelor pentru depanare (Debug Toolbar)

Bara instrumentelor pentru depanarea aplicației se accesează prin clic pe butonul drept de mouse în zona barelor cu instrumente VB și selectarea opțiunii *Debug*. Este esențială reținerea tastelor pentru activarea funcțiilor principale.

Există butoanele *Rulează/Continuă* (*Run/Continue* - **F5**), *Pauză* (*Pause* - **Ctr+Break**) și *Terminare* (*End*). Din lista butoanelor vor fi validate numai cele care pot fi folosite în contextul curent (mai sus aplicația nu este pornită, obesrvați că butoanele **Pause** și **End** sînt inactive).

Simbolul grafic de mînă (**F9**) basculează starea unui punct de întrerupere între activa și inactiv cum s-a deiscutat deja.

Păşeşte În (*Step Into* - **F8**), *Păşeşte Peste* (*Step Over* - **Shift+F8**) şi *Păşeşte Afară* (**Ctrl+Shift+F8**) se folosesc pentru a controla execuția codului linie cu linie după cum se discută în pagina următoare.

Fereastra Imediată (*Immediate Window* - **Ctrl+G**) se poate folosi pentru rularea unui cod simplu în timpul depanării, inclusiv pentru citirea/scrierea datelor, apelul de proceduri și testarea noului cod. Fereastra valorilor locale (*Locals Window*) și cea de monitorizare (*Watch Window*) se folosesc pentru monitorizarea valorilor locale sau a celor globale. *Locals Window* permite modificarea valorilor locale, iar *Watch Window* permite realizarea unor monitorizări

și implementarea unor puncte de întreuprere bazate pe valorile unor variabile din aplicație.

Fereastra de Monitorizare Rapidă (*Quick Watch* - **Shift+F9**) pune o variabilă selectată sau o expresie în *Watch Window* (*Fereastra de Monitorizare*)

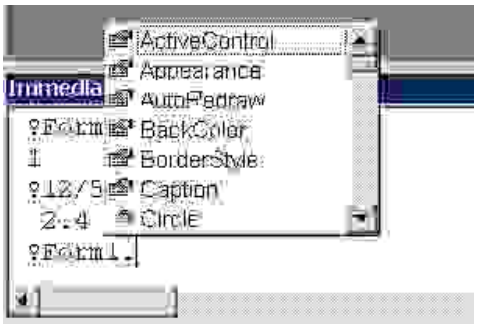
Fereastra de vizualizare a apelurilor (*View Calls* - **Ctrl+L**) vizualizează istoricul apelurilor de proceduri plecînd de la cea curentă înapoi la rutina de tratare a evenimentului care a fost declanșată pentru a vedea modul în care a rulat aplicația din acel moment.

Păşirea prin cod (rularea linie cu linie)

- **F8 - Step Into (Păşire În)**
  - **se rulează o singură linie**
  - **intră în fiecare apel de procedură**
- **Shift+F8 - Step Over (Păşire Peste)**
  - **se rulează o singură linie**
  - **procedurile se rulează complet, fără întrerupere**
- **Ctrl+Shift+F8 - Step Out (Păşire în Afară)**
  - **codul se execută din punctul curent**
  - **se opreşte după ieşirea din procedura curentă**
- **Ctrl+F8 - Run to Cursor (Rulare pînă la cursor)**
  - **punct de întrerupere temporar**

Immediate Window (fereastra imediată)

- **cunoscută și sub denumirea de Debug Window (Fereastra de depanare)**
- **deschidere cu Ctrl+G**
- **se scrie 1 singură linie de instrucțiune**
- **din cod, afișarea în ea se face cu `Debug.Print`**



10. Fereastra imediată

Immediate Window era cunoscută sub numele de Debug Window în versiunile mai vechi de VB. Se deschide cu **Ctrl+G** şi lucrează cu metodele **`Print`** şi **`Assert`** ale obiectului **`Debug`**.

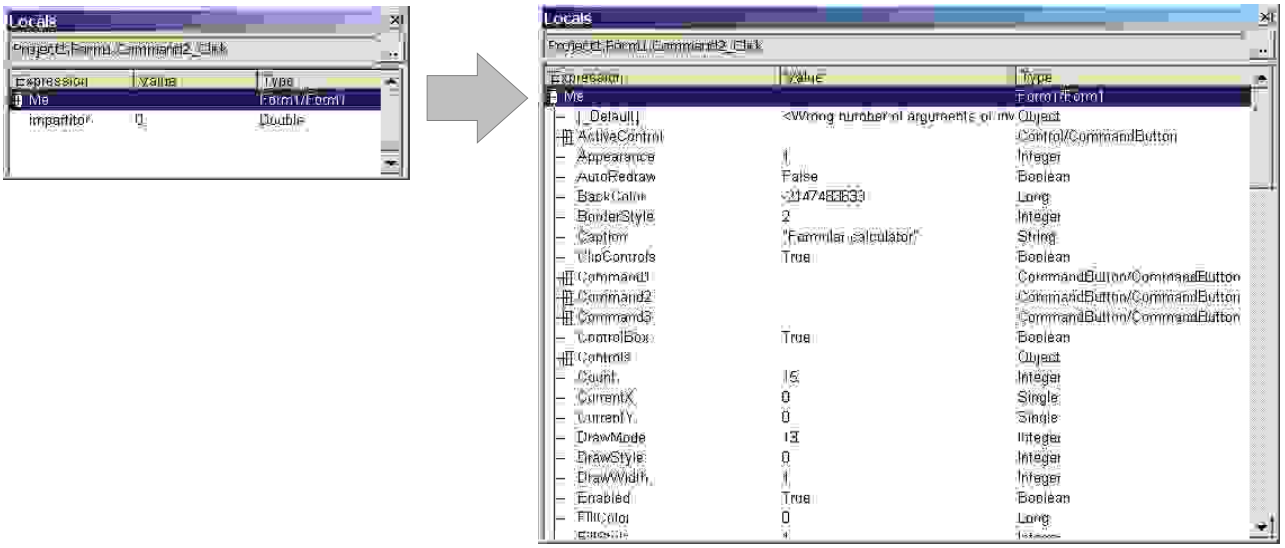
`Debug.Assert` permite punerea programului în modul de întrerupere pe baza rezultatului uni test.

`Debug.Print` ***`expresie`*** va afişa valoarea lui ***`expresie`*** direct în *Immediate Window*. Este utilizată pentru afişarea unor sevenţe de mesaje legate de starea aplicaţiei. În *Immediate Window* **`?`** este echivalentul lui **`Debug.Print`** din aplicaţie.

Aproape orice instrucţiune VB poate fi scrisă în fereastră cu meţiunea că ea trebuie să poată fi executată într-o singură linie de cod. În cazul obiectelor facilitaţile *IntellSense* şi *AutoComplete* sînt active.

Locals Window

- **permite vizualizarea şi editarea variabilelor locale**
- **asigură accesul la `Me`**



11. Fereastra variabilelor locale (Locals Window)

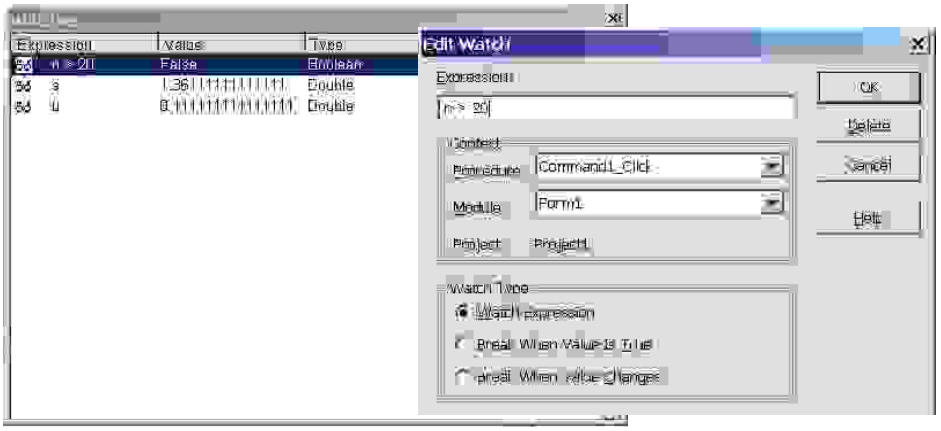
Locals window permite vizualizarea tutror variabilelor şi parametrilor care sînt vizibile într-o procedură.

Valorile se vizualizează, iar la clic dublu se afişează o fereastră de editare în care se poate introduce o nouă valoare.

Cea ma impresionantă dintre facilităţi este cea de afişare a referinţei **`Me`**. **`Me`** este o referinţă la obiectul în curs de rulare, în cazul de mai sus este chiar formularul în cauză. Ea se poate expanda pentru a avea acces la toate variabilele şi proprietăţile declarate ale formularului, fiecare control cu propriile lui proprietăţi etc. Toate acestea vor putea fi editate cu excepţia cazului în care sînt nemodificabile (read-only) sau dacă necesită repornirea proiectului.

Watch Window (Fereastra de monitorizare)

- **Vizualizează valori de variabile, obiecte sau expresii**
- **Întreruperea poate fi condiționată:**
  - **cînd condiția este `True`**
  - **cînd expresia se modifică**



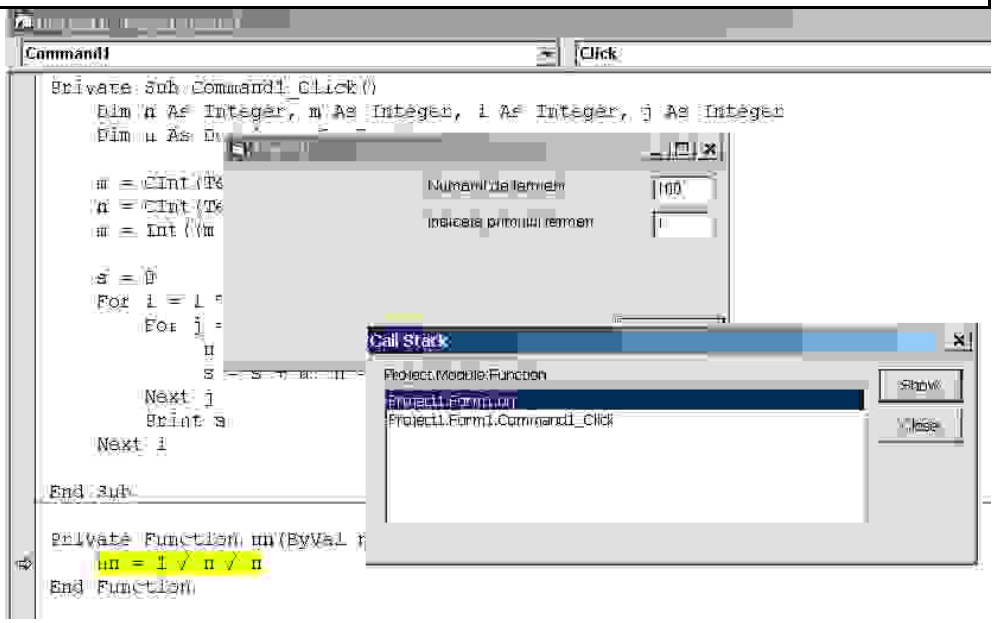
12. Fereastra de monitorizare (Watch Window)

Watch Window poate fi folosită în următoarele feluri:

- cel mai de se foloseşte pentru vizualizarea valorilor unei variabile sau expresii. Variabila sau proprietatea de urmărit se va selecta din cod şi va fi trasă şi lăsată în *Watch Window*. Acelaşi efect se obține prin clic pe butonul drept de mouse, după care se va selecta **Add Watch** ...
- fereastra are opțiunea de *Tip Urmărire* (**Watch Type**) care asigură un alt mod de utilizare - monitorizarea unei variabile **`Boolean`** sau a unei expresii; codul se rulează atît timp cît expresia monitorizată este **`False`**. Este utilă atunci cînd un număr mare de operații trebuie să fie îndeplinite înainte de a putea începe depanarea.
- o altă utilizare este aceea în care programul este oprit atunci cînd valoarea unei variabile sau expresii se modifică. Împreună cu *View Calls* ne permite să determinăm de ce se modifică valoarea unei variabile atunci cînd nu ar trebuie să fie aşa.

View Calls (Fereastra de vizualizare a apelurilor)

- **Afișează istoricul apelurilor de pe stivă**



13. Fereastra de vizualizare a apelurilor (View Calls)

Butonul **View Calls** se găseşte pe bara instrumentelor de depanare (Debug Toolbar) sau din meniul *View* selectaţi *Call Stack*.

Ea afişează istoria apelurilor de procedură plecînd de la procedura curentă înapoi spre rutina de tratare a evenimentului care a iniţiat secvenţa de apeluri.

Se foloseşte pentru a determina modul în care s-a ajuns la apelarea unei proceduri deseori împreună cu opţiunea **Break When Value Changes** din **Watch Window**.