

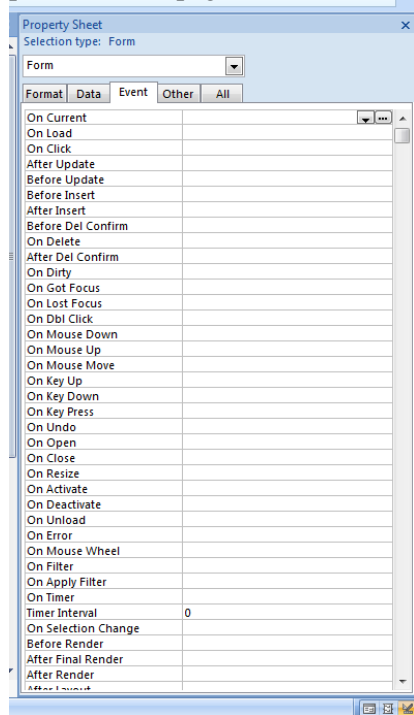
Curs 7. MODELUL ORIENTAT PE EVENIMENTE

1. Tipuri de evenimente în Access
2. Ordinea producerii evenimentelor în Access
3. Tratarea evenimentelor
5. Macrocomenzi speciale
6. Rularea și depanarea macrocomenzilor

1. Tipuri de evenimente în Access

Microsoft Access permite dezvoltarea de aplicații orientate pe evenimente. Un eveniment este o acțiune pe care aplicația o recunoaște (cum ar fi apăsarea unei taste, efectuarea unui clic cu mouse-ul, deschiderea unui formular sau raport, etc.) și care generează un anumit răspuns. Răspunsul poate fi executarea unei macrocomenzi, a unei proceduri VBA sau evaluarea unei expresii. Evenimentele sunt declanșate de acțiunile utilizatorului. Obiectele supuse evenimentelor sunt formularele, rapoartele și controalele acestora. Astfel, aplicațiile Access trebuie să fie proiectate în așa fel încât să răspundă evenimentelor declanșate de utilizator.

Pentru a răspunde unui anumit eveniment, trebuie să îi atribuim o macrocomandă, o procedură sau o expresie. Acest lucru îl putem face în pagina *Event* a ferestrei *Properties* a obiectivului respectiv



Ordinea în care se execută codul aplicației este astfel determinată de ordinea în care sunt invocate evenimentele prin acțiunile utilizatorului. Aceasta este esența programării orientate pe evenimente: utilizatorul este cel care efectuează anumite acțiuni, iar aplicația răspunde în consecință, prin intermediul codului scris de programator.

Exemplu: putem presupune că utilizatorul va trebui să introducă o valoare într-o casetă de text înainte de a pune să apese pe un buton. Pentru aceasta, dezactivăm butonul respectiv înainte ca utilizatorul să introducă o valoare în caseta text.

Access îi pune programatorului la dispoziție mai multe tipuri de evenimente ce pot fi declanșate de acțiunile utilizatorului și tratate ca aplicație.

1.1. Evenimente generate de accesarea datelor

Aceste evenimente au loc ori de câte ori utilizatorul adaugă, modifică sau șterge datele dintr-un formular ori control sau când trece de la o înregistrare la alta.

- **On Current** – Are loc la deschiderea unui formular sau când se trece de la o înregistrare la alta.
- **On Delete** – Are loc atunci când utilizatorul efectuează o operație de ștergere a datelor, înainte ca datele să fie efectiv șterse. Procedura pentru tratarea acestui eveniment are un parametru: *Cancel* care, dacă are valoarea *True*, ștergerea nu va fi efectuată, iar dacă are valoarea *False*, vor avea loc următoarele evenimente, în ordine: *Delete*, *Current* (pentru accesarea înregistrării următoare),

BeforeDelConfirm, afișarea unei casete de mesaj pentru confirmarea ștergerii, *AfterDelConfirm*.

- **BeforeDelConfirm** – Are loc după ce utilizatorul a șters una sau mai multe înregistrări și înainte ca sistemul să afișeze caseta de mesaj pentru confirmare. Procedura pentru tratarea acestui eveniment are doi parametri: *Cancel* și *Response*. Dacă parametrul *Cancel* are valoarea *True*, înregistrările nu vor mai fi șterse și caseta de mesaj nu va mai apărea. Dacă el are valoarea *False*, iar parametrul *Response* are valoarea 0, ștergerea va fi efectuată fără a mai apărea caseta de mesaj, iar dacă *Response* are valoarea 1, caseta de mesaj va apărea.
- **AfterDelConfirm** – are loc după confirmarea și/sau ștergerea înregistrărilor. Procedura pentru tratarea acestui eveniment are un argument, *Status*, care poate avea valorile 0, 1 sau 2. Valoarea 0 spune că ștergerea a fost efectuată cu succes, 1 că Access a anulat-o și 2 că utilizatorul a anulat-o.
- **BeforeInsert** – Are loc atunci când utilizatorul introduce primul caracter al unei înregistrări noi, dar înainte ca înregistrarea să fie inserată în tabelă. Procedura pentru tratarea acestui eveniment are un argument, *Cancel*, care, dacă are valoarea *True*, acțiunea de inserare va fi abandonată.
- **AfterInsert** – Are loc după inserarea unei înregistrări noi într-o tabelă.
- **BeforeUpdate** – Apare înainte ca datele dintr-un control sau dintr-o înregistrare să fie modificate efectiv. Procedura pentru tratarea acestui eveniment are un argument, *Cancel*, care, dacă are valoarea *True*, modificarea nu se va mai efectua.

- **AfterUpdate** – Are loc după ce au fost modificate datele dintr-un control sau dintr-o înregistrare. Controalele atașate au o anumită proprietate, *OldValue*, care păstrează valoarea controlului dinaintea modificării până după ce acest eveniment a avut loc. Astfel, aveți posibilitatea ca în procedura pentru tratarea acestui eveniment să redați controlului vechea valoare astfel:

```
FORMS! Formular!control = FORMS! Formular!control.OldValue
```

- **Change** – Apare atunci când se efectuează o modificare într-o casetă de text sau în câmpul de editare al unei casete combinate.
- **NotInList** – Este un eveniment specific casetelor combinate și poate apărea numai dacă proprietatea *LimitToList* a unui astfel de control are valoarea *Yes*. Astfel, evenimentul are loc dacă utilizatorul introduce în câmpul de editare al unei casete combinate o valoare care nu se regăsește în lista acestuia. Procedura pentru tratarea acestui eveniment are două argumente: *NewData*, care păstrează valoare nou introdusă de utilizator și *Response*, care poate avea valorile 0, 1 sau 2. Valoarea 0 îi indică lui Access să afișeze un mesaj standard pentru a înștiința utilizatorul că a introdus o valoare care nu se află în lista casetei combinate, 1 îi indică lui Access să nu afișeze mesajul standard și nici să nu introducă valoare în listă (dându-vă posibilitatea de a afișa un mesaj propriu), iar valoarea 2 îi indică lui Access să nu afișeze mesajul standard și să introducă noua valoare în listă. În acest ultim caz, va trebui să introduceți noua valoare și în sursa de date a casetei combinate.
- **Update** – Are loc atunci când au fost modificate datele unui obiect OLE și este specific controalelor cadru pentru obiecte OLE atașate sau neatașate. Procedura pentru tratarea acestui eveniment are un parametru, *Code*, care poate avea valorile 0, 1, 2 sau 3. Valoarea 0 îi indică lui Access că datele obiectului au fost modificate, 1 că datele au fost salvate de aplicația care a creat obiectul, 2 că fișierul ce conține obiectul OLE a fost închis de aplicația care l-a creat, 3 că fișierul ce conține obiectul a fost redenumit de aplicația care l-a creat.

1.2. Evenimente legate de focus

Apar atunci când un control, formular sau raport primește sau pierde focusul (sau când devine activ sau inactiv).

- **Enter** – Are loc înainte ca un control să primească focusul de la un alt control al aceluiași formular sau la deschiderea formularului, pentru primul control al acestuia. Evenimentul apare înaintea lui *GotFocus* și după *Current*.
- **Exit** – Are loc înainte ca un control să piardă focusul în favoarea altui control de pe formular, înaintea evenimentului *LostFocus*.

Notă: Spre deosebire de evenimentele *GotFocus* și *LostFocus*, evenimentele *Enter* și *Exit* nu vor avea loc dacă focusul trece la un alt formular sau raport. Ele pot fi folosite pentru a afișa mesaje înaintea actualizării controalelor sau pentru a schimba ordinea în care se trece cu tasta Tab de la un control al formularului la altul.

- **GotFocus** – Apare atunci când un control de pe un formular primește focusul sau când un formular primește focusul, dacă toate controalele lui sunt dezactivate. Un control nu poate primi focusul dacă este dezactivat sau dacă nu este vizibil. Acest eveniment are loc după evenimentul *Enter*.
- **LostFocus** – Are loc atunci când un formular sau un control de pe un formular pierde focusul, după apariția evenimentului *Exit* (care poate să aibă loc sau nu).
- **Activate** – Are loc atunci când un formular sau un raport primește focusul și devine activ. Aceasta se întâmplă la deschiderea formularului sau a raportului, când se face clic pe un control al său sau când este apelată procedura VisualBasic *SetFocus*. Formularul sau raportul trebuie să fie vizibil pentru ca evenimentul să aibă loc.
- **Deactivate** – Apare atunci când un formular sau raport pierde focusul în favoarea altei ferestre (cum ar fi fereastra Database, fereastra unei tabele, a unei macrocomenzi, a unui modul sau alt formular sau raport). Acest eveniment nu are loc dacă focusul trece către o cutie de dialog sau către altă aplicație.

1.3. Evenimentele legate de tastatură

Acestea au loc la apăsarea unei taste sau ca rezultat al instrucțiunii *SendKeys*.

- **KeyDown** – Apare de câte ori este apăsată o tastă atunci când un control sau un formular are focusul.
- **KeyUp** – Apare de câte ori o tastă care a fost apăsată se relaxează (când un control sau un formular are focusul).

Procedurile ce tratează acest eveniment au două argumente: *KeyCode* și *Shift*. *KeyCode* are valoare un întreg ce reprezintă tasta care a fost apăsată. Valoarea 0 nu corespunde nici unei taste. Argumentul *Shift* are ca valoare un întreg ce arată dacă tastele Shift, Ctrl sau Alt au fost apăsată în combinație cu alte taste. Valoarea 0 arată că aceste taste nu au fost apăsată, 1 că a fost apăsată tasta Shift, 2 pentru Ctrl, 4 pentru Alt. Dacă este apăsată o combinație a acestor taste, valoare parametrului *Shift* este suma valorilor corespunzătoare.

- **KeyPress** – Apare atunci când o tastă sau o combinație de taste ce corespund unui caracter ce poate fi tipărit este apăsată și apoi relaxată, pentru un control sau formular. El nu are loc la apăsarea tastelor funcționale (F1 – F12), a tastelor pentru navigare (săgeți, Home, End, Page Up, page Down) sau Shift, Ctrl și Alt. Procedura pentru tratarea acestui eveniment are un argument, *KeyAscii*, ce are ca valoare un întreg reprezentând codul caracterului respectiv.

Notă: Folosiți aceste evenimente cu precauție, deoarece dacă o tastă este ținută apăsată, evenimentele *KeyDown* și *KeyPress* au loc repetat, ceea ce poate duce la terminarea resurselor sistemului. Pentru a urmări acțiunile utilizatorului, puteți folosi în schimb evenimentele generate de accesarea datelor, cum ar fi *Change* și *Updated*.

1.4. Evenimente legate de activitatea cu mouse-ul

Apar atunci când se efectuează anumite acțiuni cu mouse-ul asupra unui control sau formular;

- **Click** – Are loc atunci când se efectuează un clic cu mouse-ul pe un formular sau pe un control ori pe o secțiune a unui formular. Atunci când se face clic pe un control aflat în cadrul unui grup de opțiuni, acest eveniment nu este al controlului respective, ci al grupului. Acest eveniment mai apare și în următoarele situații (când nu se face clic cu mouse-ul):
 - când este selectată o valoare din lista unei casete combinate cu ajutorul săgeților și este apăsată tasta Enter pentru a plasa valoarea în câmpul de editare al casetei combinate;
 - dacă este apăsată tasta Space atunci când o casetă de validare, un buton de comandă sau un buton de opțiune are focusul;
 - dacă se apasă tasta Enter pentru un formular care are un buton a cărei proprietate Default are valoarea Yes sau dacă se apasă tasta Esc pentru un formular cu un buton a cărui proprietate Cancel are valoarea Yes;
 - dacă se accesează un buton de comandă cu ajutorul comenzii sale prescurtate (Alt + litera subliniată din titlul butonului).
- **DbClick** – Are loc atunci când se face clic de două ori la rand cu mouse-ul pe un formular, pe un control al lui sau pe o secțiune a acestuia. Atunci când se face clic pe un control aflat în cadrul unui grup de opțiuni, acest eveniment nu este al controlului respectiv, ci al grupului. Procedura pentru tratarea acestui eveniment are un argument, *Cancel*, care, dacă are valoarea True, evenimentul este anulat.
- **MouseMove** – Apare atunci când utilizatorul mișcă mouse-ul. Procedura pentru tratarea acestui eveniment are patru argumente: *Button* (care arată care dintre butoanele mouse-ului a fost apăsat), *Shift* (arată dacă au fost apăstate tastele Shift, Ctrl sau Alt), *X* și *Y* (care dau coordonatele curente ale cursorului mouse-ului). Argumentul *Button* poate lua valorile: 0 dacă nu a fost apăsat nici unul dintre butoanele mouse-ului, 1 dacă a fost apăsat butonul din stânga, 2 pentru cel din dreapta și 4 pentru cel mijlociu. Dacă au fost apăstate mai multe butoane o dată, valoarea lui *Button* va fi suma valorilor corespunzătoare. Valorile argumentului *Shift* sunt aceleași ca la procedurile pentru tratarea evenimentelor de la tastatură.
- **MouseDown** și **MouseUp** – Au loc atunci când un buton al mouse-ului este apăsat sau relaxat, în timp ce cursorul se află deasupra unui formular, control al lui sau secțiune a acestuia. Procedura pentru tratarea acestui eveniment are aceleași argumente ca și cea pentru tratarea lui *MouseMove*, cu diferența că argumentul *Button* poate avea valorile: 1 (pentru butonul din stânga), (pentru butonul din dreapta) sau 4(pentru butonul din mijloc). Dacă au fost apăstate mai multe butoane o dată, vor avea loc tot atâtea evenimente *MouseDown* și *MouseUp*.

1.5. Evenimente legate de tipărire

Au loc pentru fiecare secțiune a unui raport atunci când acesta este tipărit sau este formatat pentru a fi tipărit.

- **Format** – Apare înainte ca Access să formateze fiecare secțiune a unui raport, dar după ce datele au fost selectate. Pentru secțiunea Detail, acest eveniment se produce pentru fiecare înregistrare în parte. Procedura pentru tratarea acestui eveniment are două argumente: *Cancel* și *FormatCount*. Dacă argumentul *Cancel* are valoarea True, se renunță la formatarea secțiunii curente și se trece la următoarea secțiune. *FormatCount* reprezintă numărul de evenimente *Format* produse pentru secțiunea curentă.
- **Retreat** – Apare atunci când Access trebuie să revină la o secțiune anterioară a unui raport în timpul formătărilor. El are loc după evenimentul *Format* și înaintea evenimentului *Print*, pentru a vă da posibilitatea de a modifica formătărilor făcute deja.
- **Print** – Are loc după formatare și înainte de afișare sau tipărire. Ca și evenimentul *Format*, acest eveniment are loc pentru fiecare secțiune în parte. Procedura de tratare are două argumente: *Cancel* și *PrintCount*. Dacă argumentul *Cancel* are valoarea True, secțiunea sau înregistrarea curentă nu va mai fi tipărită, *PrintCount* reprezintă numărul de apariții ale acestui eveniment pentru înregistrarea curentă. Astfel, puteți afla dacă o înregistrare va fi tipărită pe mai mult de o pagină și puteți renunța la ea.

1.6. Evenimentele ferestrelor

Acestea se produc atunci când o fereastră a unui formular sau raport este deschisă, redimensionată sau închisă.

- **Open** – Se produce la deschiderea unui formular sau raport și înainte ca prima înregistrare a formularului să fie afișată sau ca raportul să fie afișat sau tipărit. Procedura pentru tratarea evenimentului are un argument, *Cance*, care, dacă are valoarea *True*, formularul sau raportul respectiv nu va mai fi deschis. Acest eveniment se produce înaintea evenimentului *Load*.
- **Close** – Are loc atunci când un formular sau un raport este închis sau nu mai este vizibil pe ecran, după evenimentul *Unload*.
- **Load** – Apare la deschiderea unui formular și la afișarea înregistrărilor, înaintea evenimentului *Current* al primei înregistrări sau al primului control al formularului, dar după evenimentul *Open*.
- **Unload** – Apare la închiderea unui formular (dar înainte ca acesta să dispară de pe ecran), înaintea evenimentului *Close*. Procedura pentru tratarea evenimentului are un argument, *Cancel*, care, dacă are valoarea *True*, formularul nu va mai fi închis.

Notă: Atenție! Dacă dați argumentului *Cancel* valoarea *True*, formularul nu va mai fi închis sau șters de pe ecran. Astfel, dacă nu îi dați explicit mai târziu valoarea *False*, nu veți mai putea închide formularul decât închizând aplicația.

- **Resize** – Apare la deschiderea unui formular sau redimensionarea sa. Astfel, aveți posibilitatea să redimensionați și controalele formularului.

1.7. Evenimente generate de erori

- **Error** – Are loc atunci când apare o eroare de execuție la rularea aplicației. Astfel, aveți posibilitatea de a intercepta mesajele de eroare ale lui *Access* și de a afișa propriile dumneavoastră mesaje. Procedura pentru tratarea evenimentului are două argumente: *DataErr*, care reprezintă codul de eroare returnat de funcția *Err* ce se apelează de câte ori apare o eroare și *Response*, care determină dacă trebuie să fie afișat un mesaj de eroare standard. Dacă *Response* are valoarea 0, *Access* nu va mai afișa mesajul de eroare, putând astfel să afișați un mesaj personalizat, iar dacă are valoarea 1, *Access* va afișa mesajul standard.

1.8. Evenimente legate de timer

- **Timer** - Apare la intervale de timp regulate, dacă ați stabilit proprietatea *TimerInterval* a unui formular. Dacă această proprietate are valoarea 0, evenimentul nu se va produce. Pentru valori cuprinse între 0 și 65536, evenimentul va avea loc la intervalul stabilit, valoarea reprezentând mărimea intervalului în milisecunde.

2. Ordinea producerii evenimentelor în Access

2.1. Ordinea producerii evenimentelor legate de controale

Când un control primește focusul, au loc următoarele evenimente:

Enter → *GotFocus*

Când introducem sau modificăm datele dintr-un control și apoi trecem focusul altui control, ordinea producerii evenimentelor este următoarea:

KeyDown → *KeyPress* → *Change* → *KeyUp* → *BeforeUpdate* → *AfterUpdate* → *Exit* → *LostFocus*

Notă: Dacă în câmpul de editare al unei casete combinate este introdusă o valoare care nu se regăsește în lista acestuia, după evenimentul *KeyUp* apar evenimentele *NotInLst* și *Error*.

2.2. Ordinea producerii evenimentelor legate de lucrul cu înregistrările unui formular

Evenimentele ce se produc atunci când sunt afișate înregistrările unui formular, sunt diferite de cele ale controalelor care afișează date ale înregistrărilor respective.

Dacă modificăm o înregistrare prin intermediul controalelor unui formular și trecem apoi la următoarea înregistrare, se vor produce, în ordine, următoarele evenimente:

Current(formular) → *Enter(control)* → *GotFocus(control)* → *BeforeUpdate (control)* → *AfterUpdate(control)* → *BeforeUpdate(formular)* → *AfterUpdate(formular)* → *Exit(conrol)* → *LostFocus(control)* → *Current(formular)*

La ștergerea unei înregistrări, ordinea producerii evenimentelor este următoarea:

Delete → BeforeDelConfirm → AfterDelConfirm

Când trecem focusul pe o înregistrare nouă (ce nu conține încă date), vor apărea evenimentele:

Current(formular) → Enter(formular) → GotFocus(control) → BeforeInsert(formular) → AfterInsert(formular)

2.3. Ordinea producerii evenimentelor legate de formulare

Lucrul cu un formular implică generarea unor evenimente legate de închidere, deschidere, trecerea de la un formular la altul și lucrul cu datele formularului.

Exemplu: dacă deschidem formularul f1, apăsați pe un buton al său pentru a deschide formularul f2 și apoi treceți din nou pe f1, vor avea loc, în ordine, evenimentele:

Open(f1) → Load(f1) → Resize(f1) → Current(f1) → Open(f2) → Load(f2) → Resize(f2) → Deactivate(f1) → Activate(f2) → Current(f2) → Deactivate(f2) → Activate(f1)

Observăm că evenimentul Deactivate al formularului f1 apare după evenimentele Open, Load și Resize ale lui f2. Astfel, dacă în timpul deschiderii celui de-al doilea formular apare o eroare, îl putem închide și pune focusul de pe primul formular.

Evenimente ale controalelor subformularului → Evenimente ale controalelor formularului → Evenimentele formularului → Evenimentele subformularului

2.4. Ordinea producerii evenimentelor legate de tastatură și mouse

Aceste evenimente apar atunci când un formular sau un control al unui formular primește focusul. Ordinea producerii evenimentelor generate de apăsarea unei taste este:

KeyDown → KeyPress → KeyUp

Ordinea evenimentelor generate de apăsarea unui buton al mouse-ului este:

MouseDown → MouseUp → Click

Dacă se efectuează un dublu clic, evenimentul DblClick va apărea după evenimentul Click.

2.5. Ordinea producerii evenimentelor rapoartelor

Evenimentele unui raport sunt generate de tipărirea, afișarea sau de închiderea raportului respectiv. Astfel, când deschideți un raport pentru a-l tipări sau afișa și apoi îl închideți, vor apărea, în ordine, evenimentele:

Open → Activate → Format → Print → Close → Deactivate

Evenimentul *Open* are loc înainte ca interogarea pe care se bazează raportul să fie executată. Dacă interogarea nu generează nici o înregistrare, se produce evenimentul *NoData*.

3. Tratarea evenimentelor

După ce am decis ce acțiuni dorim să fie efectuate ca răspuns la anumite evenimente, nu mai trebuie decât să scriem codul VisualBasic corespunzător procedurilor pentru tratarea evenimentelor respective. Făcând clic pe săgeata din dreapta câmpului unui eveniment, se va deschide o listă din care vom putea alege o macrocomandă. Pentru a crea procedura pentru tratarea unui eveniment și a-i scrie codul VBA, procedăm astfel:

1. Deschidem formularul sau raportul în modul *Design*. Dacă doriți să tratați un eveniment al unui control sau al unei secțiuni, selectați controlul sau secțiunea respectivă.
2. Deschidem fereastra *Properties* a formularului, raportului, secțiunii sau controlului respectiv și selectăm pagina *Event*.
3. Din lista corespunzătoare evenimentului pe care dorim să îl tratăm, alegem opțiunea [Event Procedure] și apăsați butonul Build (...) pentru a deschide fereastra modulului în care se va afla procedura.
4. Access va scrie automat scheletul procedurii, care va arăta astfel:

```
Sub nume_procedura ( )
End Sub
```

5. Se compilează procedura.
6. Salvăm modulul.

Astfel, de câte ori va avea loc evenimentul respectiv, această procedură va fi apelată și executată.

În acest curs am insistat mai mult pe folosirea procedurilor VBA decât a macrocomenzilor pentru tratarea evenimentelor. Aceasta pentru că procedurile sunt mai rapide în execuție, mai flexibile și mai ușor de întreținut. Spre deosebire de macrocomenzi, ele vă dau posibilitatea de a intercepta și a trata erorile cu ajutorul evenimentului Error și a funcției OnError. Cu toate acestea, există și câteva macrocomenzi care nu au corespondent în VBA: *AutoKeys*, *Autoexec* și *AddMenu*.

4. Macrocomenzi

Macrocomenzile reprezintă o metodă simplă de a efectua anumite acțiuni într-o aplicație Access, fără a avea prea multe cunoștințe de programare. Mai precis, o macrocomandă este o însușire de acțiuni, programată să execute în cazul producerii unui anumit eveniment.

Macrocomenzile sunt, așadar, niște instrumente utile, care, ne permit să programăm acțiuni simple, de la deschiderea și închiderea unui formular ori raport sau stabilirea unor opțiuni, până la emiterea unui semnal sonor. Pe de altă parte, macrocomenzile au dezavantajele lor, care îi determină pe cei mai mulți preprogramatori Access să prefere modulele și codul VBA. Access vă dă chiar posibilitatea de a transforma o macrocomandă în codul VBA echivalent. Pe lângă faptul că macrocomenzile sunt mai lente în execuție decât procedurile, ele mai au dezavantajul că nu oferă programatorului posibilitatea de a intercepta și trata corespunzător erorile.

Există însă unele acțiuni care pot fi efectuate numai prin intermediul macrocomenzilor.

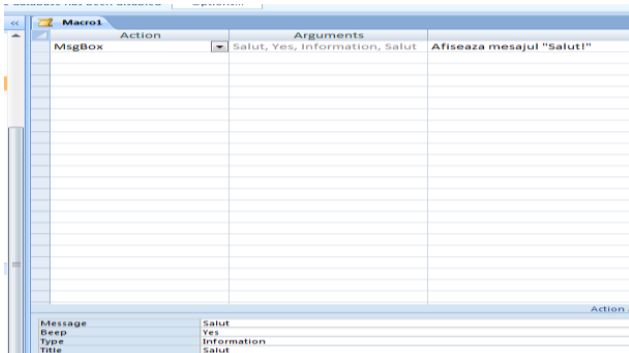
- Definirea comenzilor rapide de la tastatură pentru diferite acțiuni. *Exemplu:* dacă vom dori ca tasta F3 să deschidă un anumit formular, nu vom putea stabili acest lucru decât cu ajutorul macrocomenzii *AutoKeys*.
- Rularea anumitor acțiuni la deschidere a bazei de date.
- Crearea și utilizarea meniurilor și a barelor cu instrumente personalizate.

4.1. Crearea unei macrocomenzi

Vom descrie în cele ce urmează pașii necesari creării unei macrocomenzi care să afișeze o casetă de mesaj.

1. Create, *Macro*. Se va deschide fereastra *Macro Builder*.
2. Dăm clic pe prima linie a coloanei Action și apoi apăsăm pe săgeată. Se va deschide o listă cu toate acțiunile. Selectați acțiunea *MsgBox*.
3. În partea de jos a ferestrei Macro Builder vor apărea argumentele corespunzătoare acțiunii respective. În acest caz, ele sunt: *Message*(mesaj), *Beep* (semnal sonor), *Type* (tipul casetei de mesaj) și *Title* (titlul casetei de mesaj).
4. Salvăm macrocomanda cu ajutorul *Save*.

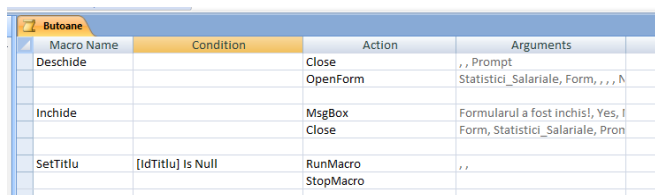
5. Pentru a rula macrocomanda, alegem *Run* sau dublu clic.



Notă: Este bine ca pentru fiecare acțiune a unei macrocomenzi să introduceți un scurt comentariu care să descrie ce anume face acțiunea respectivă. Folosiți pentru aceasta liniile coloanei Comment. Dacă veți converti ulterior macrocomanda la codul VBA corespunzător, aceste comentarii nu se vor pierde.

4.2. Grupuri de macrocomenzi

Sunt colecții de macrocomenzi simple, salvate separat, dar cuprinse în aceeași comandă globală. Fiecare macrocomandă din grup trebuie să aibă un nume propriu, pe care îl introducem în coloana *Macro Name* a ferestrei *Macro Builder*. Într-un grup, macrocomenzile componente trebuie să fie separate prin linii goale.



Macro Name	Condition	Action	Arguments
Deschide		Close	, , Prompt
		OpenForm	Statistici_Salariale, Form, , , , N
Inchide		MsgBox	Formularul a fost inchis!, Yes, I
		Close	Form, Statistici_Salariale, Pron
SetTitlu	[IdTitlu] Is Null	RunMacro	, ,
		StopMacro	

Grupurile de macrocomenzi sunt utile, de exemplu, atunci când doriți să păstrați toate macrocomenzile unui formular în același obiect, astfel încât numărul obiectelor să fie mai mic.

Pe de altă parte, o macrocomandă poate apela altă macrocomandă în cadrul unei condiții, caz în care veți prefera să le includeți pe amândouă în același obiect, pentru a fi mai simplu de depanat.

O macrocomandă din grup poate fi executată utilizând numele grupului, urmat de un punct și de numele macrocomenzii:

Nume_grup.nume_macro

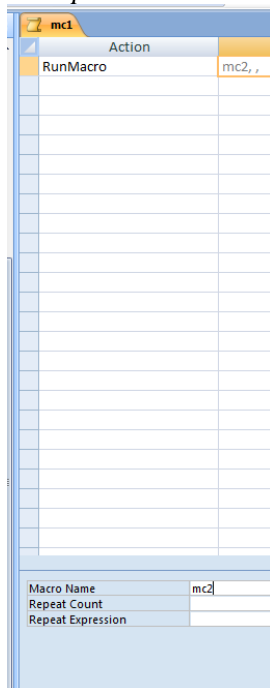
Execuția unei macrocomenzi din grup se oprește la întâlnirea următoarei linii libere din coloana *Action*.

O altă utilizare frecventă a grupurilor de macrocomenzi reprezintă crearea meniurilor personalizate, în care fiecărui control îi corespunde o macrocomandă din grup.

4.3. Macrocomenzi imbricate

În Access putem crea și macrocomenzi compuse, în care o macrocomandă să fie apelată de alta. Cel mai simplu mod de a realiza acest lucru este folosirea acțiunii *RunMacro*.

Exemplu: Macrocomanda mc1 apelează macrocomanda mc2.



Macro Name	Condition	Action	Arguments
mc1		RunMacro	mc2, ,

O altă situație în care se folosesc macrocomenzi imbricate este atunci când veți dori să controlați execuția unei macrocomenzi impunând o condiție.

4.4. Condiții în macrocomenzi

Access pune la dispoziție un mod rudimentar de a controla firul de execuție al macrocomenzilor, prin intermediul coloanei *Condition* a ferestrei *Macro Builder*.

Dacă introduceți o expresie în coloana *Condition*, acțiunea aflată pe aceeași linie în coloana *Action* se va executa numai dacă expresia este adevărată. Dacă expresia este falsă, Access nu va executa acțiunea de pe aceeași linie cu condiția și va trece la

următoarea acțiune. Dacă introduceți trei puncte de suspensie (...) în coloana *Condition* pe linia de sub o condiție existentă, Access va considera acțiunea de pe aceeași linie cu punctele de suspensie ca fiind supusă aceleiași condiții.

Exemplu: O macrocomandă supusă unei condiții este macrocomanda *Butoane.SetTitlu*. Condiția verifică dacă valoarea câmpului IdTitlu este Null și dacă da, se rulează macrocomanda NullTitlu și apoi se întrerupe execuția; altfel, se modifică valorile câmpurilor IdTitlu și Statut.

5. Macrocomenzi speciale

În cele ce urmează, vom discuta despre două macrocomenzi ce merită o atenție specială: *AutoKeys* și *AutoExec*.

5.1. Macrocomanda AutoKeys

Vă permite să atribuiți comenzi rapide de la tastatură acțiunilor pe care le poate efectua aplicația. De exemplu, ar fi util ca, de câte ori utilizatorul folosește combinația de taste Ctrl + P, să fie tipărit raportul curent. Astfel, puteți atribui orice combinație de taste oricărei acțiuni (sau grup de acțiuni) care poate fi executată prin intermediul unei macrocomenzi. Pentru a nu crea confuzii, evitați totuși atribuirea combinațiilor de taste predefinite în Access (cum ar fi Ctrl + V pentru comanda *Edit / Paste*) deoarece, în astfel de cazuri, Access va executa macrocomanda dvs. în locul comenzii predefinite.

Nu puteți atribui o macrocomandă unei singure taste alfanumerice sau unei taste folosite, în general, în combinație cu alte taste (**Ctrl** sau **Alt**). Tastelor funcționale (F1-F12) și tastelor Insert și Delete le puteți atribui macrocomenzi.

Pentru a atribui o combinație de taste unei acțiuni, procedăm astfel:

1. Se creează o nouă macrocomandă.
2. Se alege afișarea coloanei *Name* a ferestrei *Macro Builder*. Introducem aici combinația de taste pe care dorim să o atribuim.
3. Introducem în coloana *Action* corespunzătoare acțiunea atribuită combinației de taste din coloana *Name*. Dacă dorim să atribuim mai multe acțiuni unei combinații de taste, nu trebuie decât să le introducem pe linii consecutive ale coloanei *Action* (fără a lăsa linii libere între ele).
4. Inchidem macrocomanda și o salvăm cu numele : “AutoKeys”. Este important ca acesta să fie numele macrocomenzii, deoarece altfel Access nu va efectua atribuirile.
5. Inchidem baza de date și o deschidem din nou pentru a activa macrocomanda *AutoKeys*.

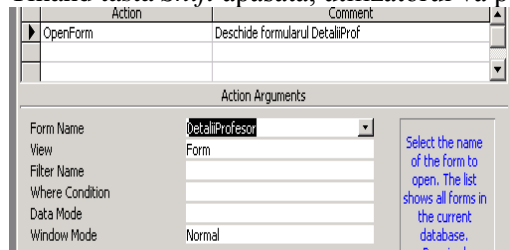
Pentru a specifica o anumită combinație de taste, trebuie să folosim o sintaxă specială, în care tastei din coloana *Tasta* îi corespund, în coloana *Name* a ferestrei *Macro Builder*, caracterele din coloana *Correspondent* a tabelului. Coloana *Exemplu* arată concret ce se introduce în coloana *Name*.

Tasta	Correspondent	Exemplu
Ctrl	^	^Z (pentru Ctrl + Z)
Tastă funcțională	{ }	{F8} (pentru tasta F8)
Shift	+	+{F8} (pentru Shift + F8)
Insert	{INS } sau {INSERT}	{INS} (pentru tasta Insert)
Delete	{DEL} sau {DELETE}	{DEL} (pentru tasta Delete)
Combinații de taste	Folosiți caracterele de mai sus	+^{F8} (pentru combinația Shift + Ctrl + F8)

5.2. Macrocomanda AutoExec

Dacă doriți ca anumite acțiuni să aibă loc ori de câte ori se deschide o bază de date, puteți crea o macrocomandă care să efectueze acțiunile respective și pe care să o salvați cu numele “AutoExec”. Cea mai frecventă utilizare a macrocomenzii *AutoExec* este ascunderea ferestrei *Database* și deschiderea unui formular care să joace rolul de meniu principal al aplicației sau de splash- screen.

Tinând tasta *Shift* apăsată, utilizatorul va putea, totuși, să împiedice rularea macrocomenzii *AutoExec* la deschiderea b.d.



6. Rularea și depanarea macrocomenzilor

Există mai multe modalități de a rula o macrocomandă în Access:

1. Făcând dublu – clic pe numele macrocomenzii.

2. Prin intermediul codului VBA dintr-un modul, folosind metoda RunMacro a obiectului DoCmd:

`DoCmd.RunMacro nume_macrocomanda`

Dintr-o altă macrocomandă sau prin intermediul barelor personalizate de meniuri sau de instrumente.

3. La deschiderea bazei de date, dacă numele macrocomenzii este AutoExec.

Vă puteți da seama dacă o macrocomandă nu funcționează când vedeți fereastra de dialog *Action Failed*. Aceasta prezintă numele macrocomenzii care a eșuat, acțiunea care a produs eroarea și argumentele acțiunii respective. Apăsați butonul **Halt** pentru a depana macrocomanda.

Putem depana o macrocomandă rulând-o acțiune cu acțiune, astfel:

1. Deschidem macrocomanda în fereastra *Macro Builder*.
2. Apăsăm butonul *Single Step* de pe bara cu instrumente sau alegeți comanda *Run / Single Step*.
3. Dacă macrocomanda începe pe prima linie a ferestrei *Macro Builder*, putem apăsa butonul *Run* de pe bară pentru a porni execuția. Dacă depănăm o macrocomandă dintr-un grup, rulăm codul care apelează macrocomanda.

Când macrocomanda începe să ruleze, pe ecran apare cutia de dialog *Macro Single Step*, care prezintă acțiunea în curs de rulare și valorile parametrilor acesteia. Această cutie de dialog ne oferă trei opțiuni: rularea acțiunii următoare (butonul *Step*), oprirea execuției macrocomenzii (butonul **Halt**) și continuarea neîntreruptă a execuției (butonul **Continue**).

Putem opri de la tastatură execuția unei macrocomenzi la un moment dat, apăsând **Ctrl + Break**. Atunci, apare fereastra *Macro Single Step*, ce permite rularea pas cu pas a macrocomenzii.