**Realizarea programelor in Visual FoxPro**

Pentru realizarea de programe in Visual FoxPro se utilizeaza comanda:

**MODIFY COMMAND (nume-fisI ?] [NOEDIT] [NOWAIT] [RANGE expN1, expN2]**

**[[WINDOW nume-fer1] [IN[WINDOW nume-fer21lN SCREEN]] [SAME] [SAVE]**

Clauzele comenzii sunt:

.nume-fis- este numele fisierului care contine programul sursa (tip .PRG);

. ? determina deschiderea ferestrei *Open* pentru selectarea fisierului sursa de editat (tip

.PRG, .MPR, .QPR, .SPR);

. NOEDIT fisierul sursa poate fi doar vizualizat, nu si editat (actualizat);

. NOWAIT la executia programului nu se asteapta daca apare fereastrade editare;

. RANGE permite selectarea pentru editare doar a unei parli (de la expNl la expN2)

din programmul sursa;

. WINDOW fereastra de editare va fi inclusa intr-o aha fereastra (nume-fer2) care

trebuie sa fie deja activata;

. IN SCREENplaseaza fereastra de editare in ecranul curent;

. SAME impiedica fereastra de editare sa devina activa;

. SAVE dupa parasirea ferestrei de editare aceasta ramane pe ecran.

Apelul unui program sursa in VFPse face prin comanda DO, care preia fisierul sursa (.PRG), il compileaza si rezulta fisierul obiect (.FXP), apoi pe acesta il link-editeaza si rezulta fisierul executabil (.EXE), pe care iI lanseaza in executie.

Comanda DO mai este folosita si pentru apelul procedurilor intr-un program apelant VFP.

Sintaxa comenzii este:

**DO numep1 [IN numep2] [WITH param]**

1. **Structurile de program**

Structurile de programare intalnite in programarea structurala si modulara se regasesc in

totalitate in sistemul VFP.

Limbajul nu are comenzi de salt neconditionat in program, avand, in schimb, comenzi specifice pentru implementarea celor trei structuri fundamentale (secvenliaIa, alternativa si repetitiva).

**1.1. Structura secventiala**

O structura seventiala (liniara) de program VFP se poate realiza printr-un grup oarecare de comenzi, dintre cele prezentate la modul de lucru cu fereastra de comanda.

In general, o astfel de structura este data de o secventa de comenzi, care nu contine structurile alternative IF si CASE si structurile repetitive (WHILE, FOR, SCAN).

**a) Comenzile pentru atribuire**

Exista doua moduri de initializare: prin comanda specifica STORE sau atribuirea propriu-zisa (=), existenta in toate limbajele de programare universale.

**STORE <expr> TO <lista-var-mem> ! <nume-tablou>**

Exemplu:

STORE 0 TO Xl, X2, X3

STORE 'TEST' TO X4

STORE .T. TO X\_LOGIC

STORE (09/10/99) TO X\_DATA

DISPLAY MEMORY LIKE X\*

**<var\_mem> = <expr>**

**<nume\_tablou> =<expr>**

Comanda = realizeaza o atribuire si totodata o evaluare. Ea are acelasi efect ca cel al comenzii STORE, cu deosebirea ca initializeaza o singura variabila si nu o lista de variabile. Variabila sau tabloul aflat in partea stanga a semnului de egalitate se initializeaza cu valoarea data de expresia din partea dreapta.

b) Comenzile pentru operatii de intrare/iesire standard (vezi carte)

**1.2. Structura alternativa**

Sunt implementate ambele forme ale structurii alternative de programarea structurata, prin urmatoarele comenzi specifice: structura alternativa cu doua ramuri (comanda IF); structura alternativa cu mai multe ramuri (comanda CASE).

A) Structura alternativa cu doua ramuri:

Structura alternativa cu doua ramuri este implementata prin comanda IFENDIF cu sintaxa urmatoare:

**IF <exprL>**

**<secventa-de-comenzi1 >**

**[ELSE**

**[secventa-de-comenzi2>]]**

**ENDIF**

Se evalueaza <exprl> si in cazul in care este adevarata se executa toate comenzile ce urmeaza pana la ELSE sau la ENDIF (daca lipseste ElSE). In cazul in care<exprl>este falsa si ElSE exista, se executa toate comenzile de la ELSE la ENDlF.

Fiecare IF trebuie sa se incheie cu un ENDIF, in caz contrar va fi afisat un mesaj de eroare la executie.

Exemplu

IF CODB>0 .AND. CODB<100000

M\_bm = M\_bm + 1

ELSE

? 'Cod beneficiar curent'

ENDIF

**b) Structura alternativa cu mai mult de doua ramuri:**

Structura alternativa cu mai mult de doua ramuri este implementata prin comanda DO CASE ENDCASE cu urmatoarea sintaxa:

**DO CASE**

**CASE <expL1>**

**<comenzi1>**

**[CASE <exprL2>**

**<comenzi2>]**

**[OTHERWISE**

**<comenzi3>]**

**ENDCASE**

Se foloseste pentru executia unui set de comenzi VFP bazat pe rezultatul unei conditii logice.

La executia unei comenzi DO CASE se evalueaza succesiv expresiile logice din structura

CASE. Rezultatul evaluarii determina care set de comenzi va fi executat.

Exemplu

DO CASE

CASE UMP = 'KG'

? 'KILOGRAME'

CASE UMP ='TONE'

? 'TONE'

CASE UMP = 'MET'

? 'METRI\_LIN'

OTHERWISE

? 'UNITATE DE MASURA ERONATA'

ENDCASE

**1.3. Structura repetitiva**

In sistemul VFP sunt implementate doua din cele trei structuri repetitive existente in programarea structurata, prin comenzi specifice si anume cele conditionate anterior. Structura repetitiva conditionata posterior, care nu are corespondent intr-o comanda VFP poate fi simulata.

Sunt implementate:

**a) structura repetitiva conditionata anterior (DO WHllE);**

**b) structura repetitiva conditionata anterior, cu numarator (FOR);**

**c) structura specifica VFP, ce parcurge o tabela de date automat de la inceput pana la sfarsit, in mod conditionat (SCAN).**

**a) Structura repetitiva conditionata anterior**

Structura este implementata prin comanda DO WHlLE ENDDO cu urmatoarea sintaxa:

**DO WHILE <expL>**

**<secventa\_de\_comenzi>**

**[LOOP]**

**[EXIT]**

**ENDDO**

Se evalueaza expresia logica <expl> si daca este adevarata, se executa setul de comenzi ce urmeaza pana la ENDDO. Fiecare DO WHILE trebuie sa se incheie cu ENDDO. Daca

<expl> devine falsa, ciclul este inchis si pointerul de control se pozitioneaza la urmatoarea comanda, ce urmeaza dupa ENDDO.

Clauza LOOP, va determina reluarea executarii comenzii DO WHlLE. Comenzile existente intre LOOP si ENDDO nu se vor mai executa.

Daca este specificata clauza EXIT, aceasta returneaza controlul programului la urmatoarea comanda ce urmeaza lui ENDDO. Aceasta reprezinta o iesire fortata din structura repetitiva.

EXIT poate apare oriunde in ciclul DO WHlLE ENDDO.

Exemplu

DO WHILE .NOT. EOF()

? DENB

IF CODB>l05

EXIT

ENDIF

ENDDO

**b) Structura repetitiva conditionata anterior, cu numarator**

Structura repetitiva conditionata anterior, cu numarator, este implementata prin comanda

FOR ENDFOR. Aceasta comanda va executa setul de comenzi de un numar de ori specificat.

Sintaxa comenzii este: .

**FOR <var>=<expl> TO <exp2> [STEP <exp3>]**

**<secventa-de-comenzi>**

**[EXIT]**

**[LOOP]**

**ENDFOR I NEXT**

unde:

. <var> este o variabila de memorie cu rol de contor (numarator). Ea determina de cate ori se executa ciclul de la FOR la ENDFOR (sau EXIT). Contorul <var> pleaca de la o valoare initiala data de <exprl> pana la o valoare finala «expr2», cu pasul dat de <expr3>. Daca pasul nu este specificat, atunci se atribuie in mod implicit valoarea 1.

Variabila <var> se comporta la fiecare intrare intr-un nou ciclu cu valoare finala <exp2>. Daca este mai mica sau egala se va executa secventa de comenzi cuprinsa intre FOR si ENDFOR. Daca variabila <var> este mai mare decat valoarea finala se executa urmatoarea comanda dupa ENDFOR.

. <exp1>, <exp2>, <exp3> sunt expresii numerice, variabile de memorie sau constante a caror valoare poate fi modificata in interiorul ciclului fara afectarea numarului specificat de cicluri. Variabila <var> va fi definita inainte de intrarea in ciclu.

Modificarea ei in interiorul ciclului va afecta numarul de cicluri. Daca <exp3> are valoare negativa, variabila de ciclare <var> va fi decrementata cu aceasta valoare la fiecare ciclu, iar valoarea initiala <expl> trebuie sa fie mai mare decat valoarea finala <exp2>.

. EXITva determina o iesire fortata din ciclul FOR.

LOOP determina reluarea comenzii FOR.Comenzile existenteintre LOOP si ENDFOR

nu se vor mai executa.

**c) Structura repetitiva SCANENDSCAN**

Este o structura repetitiva specifica sistemului VFPprin care este parcursa in mod automat o tabela. Este implementata prin comanda SCAN ENDSCAN care are urmatoarea sintaxa:

**SCAN [NOOPTIMIZE]**

**[<scop>] [FOR<expL 1>] [WHILE<expL2]**

**<secventa-de-comenzi>**

**[LOOP]**

**[EXIT]**

**ENDSCAN**

Se parcurge tabela executandu-se setul de comenzi pentru fiecare inregistrare care indeplineste conditia enuntata. Comanda SCAN incrementeaza automat pointerul de inregistrare cu 1 si testeaza conditia specificata prin clauzele FOR si WHILE.

Exemplu

SCAN

FOR CODB> 101

IF CODE = 105

LOOP

ENDIF

? DENB

ENDFOR

ENDSCAN