Capitolul 2

STRUCTURI DE DATE ŞI STRUCTURI DE PRELUCRARE

Sumar:

- Aspecte introductive privind structurile de date şi de prelucrare
- Structuri de date
- Structuri algoritmice de prelucrare

2.1 ASPECTE INTRODUCTIVE

Literatura de specialitate subliniază importanţa deosebită a înţelegerii corecte a principalelor structuri de date şi a structurilor de prelucrare folosite în activitatea de programare, precum şi în formarea specialiştilor în domeniul instrumentelor software.

O *structură de date* reprezintă o modalitate de organizare a datelor care acoperă data elementară propriu-zisă şi relaţiile acesteia cu alte date. Selectarea în activitatea de programare a unei structuri de date adecvate poate conduce la avantaje importante, între care: utilizarea eficientă a memoriei, eficientizarea prelucrărilor şi chiar reducerea costurilor de programare.

Prin *structuri de prelucrare* sau *structuri algoritmice de prelucrare* ne referim la acele modalităţi de organizare a controlului în prelucrarea prin program/ proceduri a unor structuri de date.

Este important de subliniat faptul că structurile de date sunt strâns legate de structurile algoritmice prin care se eficientizează operaţiile de creare şi actualizare a structurilor de date, dar şi operaţii de căutare şi sortare a acestora. Toate aceste structuri de date şi de prelucrare se bazează pe memoria calculatorului. În cel mai simplu mod trebuie privită *memoria* ca fiind o *structură liniară de celule/locaţii* în care se depozitează date cu valori corespunzătoare (figura 2.1).

Trebuie să facem precizarea că adresele de memorie nu se codifică după o regulă uşor de înţeles şi de utilizat în activitatea de programare. Iată, spre exemplu, o adresă de memorie: AF0021E. Acest fapt a condus la *necesitatea atribuirii de nume* în logica utilizatorilor de medii de programare şi a problemei de rezolvat.

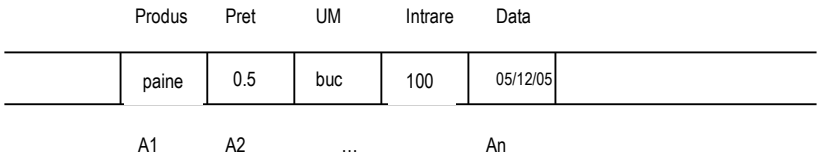


Figura 2.1 Atribuirea numelor la celulele de memorie

După o asemenea împărţire a memoriei şi atribuirea de adrese şi nume, putem stoca date în oricare din celule, în funcţie de ce ne-am propus prin numele asociat. Valorile stocate pot fi restricţionate pe categorii. De exemplu, la *Pret* nu putem introduce decât valori numerice iar la *UM* numai text. Astfel, se realizează o specializare a acestor celule în funcţie de conţinutul lor pe tipuri de valori. Această caracteristică a datei poartă numele de *tip* şi poate fi: numeric, alfabetic, alfanumeric, logic, dată calendaristică ş.a.

2.2 STRUCTURI DE DATE

2.2.1 Clasificări ale principalelor structuri de date

Compoziţia internă a unei date impune gruparea acestora în *date structurate* şi *date elementare*. O *dată elementară* se defineşte ca fiind un atribut care nu poate fi descompus din punct de vedere logic. La rândul lor, *datele structurate* sau *compuse* se definesc ca fiind o grupare de date elementare şi chiar de alte date compuse.

Trebuie precizat că toate datele în baze de date sau entităţi sunt declarate în dicţionarul de date pe ultimul nivel ca fiind date elementare.

La rândul lor, *datele structurate* se grupează în:

- *Date structurate de nivel redus*, care permit operaţii la nivel de componentă. Sunt incluse în această categorie tablourile şi articolele;
- *Date structurate de nivel înalt*, care permit operaţii implementate de algoritmi utilizator. În această categorie intră: liste liniare, liste liniare ordonate, stiva şi coada.

2.2.2 Date elementare, tablouri şi pointeri

O dată elementară reprezintă cea mai simplă modalitate de reprezentare logică a unei informaţii. În categoria datelor elementare sunt incluse:

• *Numere întregi*: au asociate celule sau locaţii de memorie în care sunt stocate *numere întregi*. Asupra acestor date se pot aplica operaţiile aritmetice cunoscute: +, -, ...;

• *Numere reale*: au asociate celule de memorie în care sunt stocate *numere raţionale*. Ca şi în cazul numerelor întregi asupra lor se pot aplica operaţiile aritmetice;

• *Valori booleene*: au asociate celule sau locaţii de memorie în care sunt stocate *valorile logice* true sau false. Aceste date fac obiectul operaţiilor/ operatorilor logici: and, or, not etc.

• *Caractere*: au asociate celule sau locaţii de memorie în care sunt stocate şiruri de caractere alfabetice de genul ‘a’, ‘b’, ‘abc’, ‘Popescu’ … . Pentru aceste date se pot aplica operaţii speciale de concatenare şi comparaţie.

În tabelul 2.1. prezentăm câteva exemple de date elementare.

Tabel 2.1. Prezentare variabile de lucru

Nr. crt.	Nume_dată	Valoare	Tip
1	Produs	Paine	Şir de caractere
2	Preţ	0.5	Numeric
3	UM	Buc	Şir de caractere
4	Intrare	100	Numeric
5	Data	05/12/05	Dată calendaristică

Particularizăm crearea datelor elementare din tabelul 2.1 în Visual Basic for Application:

```
Sub algoritm()
' creare -initializare
        produs = "paine"                ' creare variabila produs si intializare cu "paine"
        pret = 0.5                      ' creare variabila pret si intializare cu 0.5
        um = "buc"                      ' creare variabila UM si intializare cu "buc"
        intrare = 100                   ' creare variabila intrare si intializare cu 100
        data = "12/12/2005"             ' creare variabila data si intializare cu "12/12/2005"
' calcul valoare
        v = pret * intrare              ' creare variabila v si incarcare cu produsul dintre continutul
                                        ' variabilei pret cu cel al variabilei intrare
' afişare rezultate
        MsgBox "Variabila v contine: " & Str(v)                         ' Afisare variabila v
        MsgBox "Total intrari " & produs & ": " & Str(v) & " RON"       ' Afisare rezultate finisate
End Sub
```

Observaţie: semnul ' de pe linia cu instrucţiuni permite ataşarea unui comentariu.

Rezultatul execuţiei acestui *Algoritm* este cel din figura 2.2.

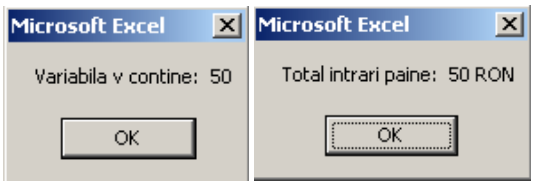


Figura 2.2 Rezultate la execuţia procedurii Algoritm

Acelaşi algoritm se putea scrie şi astfel:

```
        Public produs As String ' se creaza in mod public variabila produs de tip sir
        Public pret             ' se creaza in mod public variabila pret de tip variant
        Public um As String
        Public intrare
        Public data As Date     ' se creaza in mod public variabila produs te tip data calendaristica
        Public v
Sub algoritm()
' creare-initializare
        produs = "paine"        ' intializare cu "paine"
        pret = 0.5              ' intializare cu 0.5
        um = "buc"              ' intializare cu "buc"
        intrare = 100           ' intializare cu 100
        data = "12/12/2005"     ' intializare cu "12/12/2005"
' calcul valoare
        v = pret * intrare      ' incarcare cu produsul dintre continutul lui pret si cel al variabilei intrare
' afişare rezultate
        MsgBox "Variabila v contine: " & Str(v)                         ' Afisare variabila v
        MsgBox "Total intrari " & produs & ": " & Str(v) & " RON"       ' Afisare rezultate finisate
End Sub
```

Un *tablou* se defineşte ca fiind cea mai simplă dată structurată organizată sub formă tabelară. O caracteristică importantă a unui tablou o reprezintă dimensiunea acestuia. Un tablou cu o singură dimensiune (o singură coloană) poartă denumirea de *vector*, iar un tablou cu două dimensiuni poartă denumirea de *matrice*. În figura de mai jos reprezentăm schematic un vector şi o matrice.

Din figura următoare şi din definiţiile date trebuie să se reţină faptul că în fiecare celulă a unui tablou se regăseşte o dată elementară.

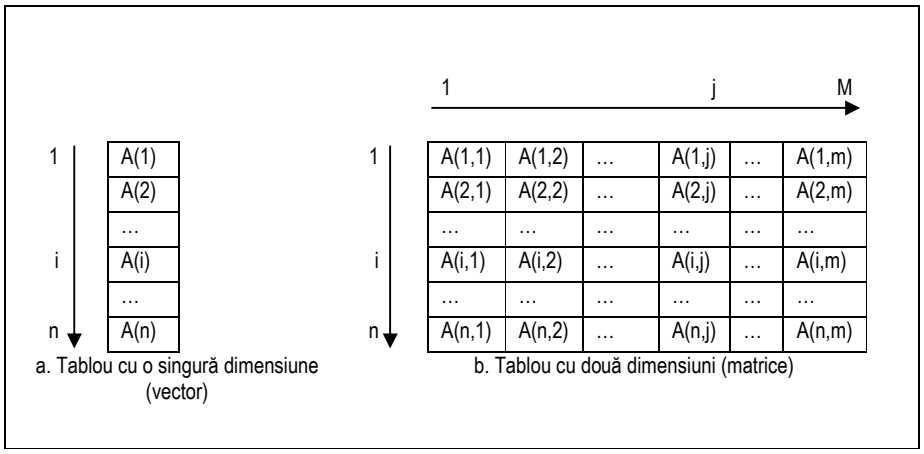


Figura 2.3 Structuri de date de tip tablou

Datele elementare, din exemplul precedent, se pot converti în date grupate sub următoarea formă, folosind un tablou cu o singură dimensiune.

```
Sub Alg_tablou_1()
' creare-initializare
        Dim aprov(5)
        aprov(0) = "paine"          ' intializare cu "paine"
        aprov(1) = 0.5              ' intializare cu 0.5
        aprov(2) = "buc"            ' intializare cu "buc"
        aprov(3) = 100              ' intializare cu 100
        aprov(4) = "12/12/2005"     ' intializare cu "12/12/2005"
' calcul valoare
        aprov(5) = aprov(1) * aprov(3)
' afisare rezultate
        MsgBox "Total intrari " & produs & ": " & Str(aprov(5)) & " RON"    ' Afisare rezultate finisate
End Sub
```

În Visual Basic se pot crea tablouri cu maximum 60 de dimensiuni. Noi vom particulariza crearea unui tablou cu două dimensiuni: 5 linii (numerotate de la 0 la 4) şi 3 coloane (numerotate de la 0 la 2). Astfel vom considera intrarile de pâine pe 3 zile.

```
Sub Alg_tablou_2()
' creare-initializare
        Dim aprov(4, 2)
        ' prima zi
        aprov(0, 0) = "paine"           ' intializare cu "paine"
        aprov(1, 0) = 0.5               ' intializare cu 0.5
        aprov(2, 0) = "buc"             ' intializare cu "buc"
        aprov(3, 0) = 100               ' intializare cu 100
        aprov(4, 0) = "12/12/2005"      ' intializare cu "12/12/2005"
        ' a doua zi
        aprov(0, 1) = "paine"           ' intializare cu "paine"
        aprov(1, 1) = 0.5               ' intializare cu 0.5
        aprov(2, 1) = "buc"             ' intializare cu "buc"
        aprov(3, 1) = 100               ' intializare cu 108
        aprov(4, 1) = "12/12/2005"      ' intializare cu "12/12/2005"
        ' a treia zi
        aprov(0, 2) = "paine"           ' intializare cu "paine"
        aprov(1, 2) = 0.5               ' intializare cu 0.5
        aprov(2, 2) = "buc"             ' intializare cu "buc"
        aprov(3, 2) = 100               ' intializare cu 116
```

```
        aprov(4, 2) = "12/12/2005"          ' intializare cu "12/12/2005"
' calcul valoare
        aprov(5, 0) = aprov(1, 0) * aprov(3, 0)
        aprov(5, 1) = aprov(1, 1) * aprov(3, 1)
        aprov(5, 2) = aprov(1, 2) * aprov(3, 2)
' afisare rezultate
        MsgBox "Intrari Luni" & produs & ": " & Str(aprov(5, 0)) & " RON"
        MsgBox "Intrari Marti" & produs & ": " & Str(aprov(5, 1)) & " RON"
        MsgBox "Intrari Miercuri" & produs & ": " & Str(aprov(5, 2)) & " RON"
End Sub
```

Pointer-ul reprezintă o legătură către o dată elementară. Cu alte cuvinte, o variabilă care conţine o *adresă* sau un *nume de variabilă* pentru o altă dată elementară. Schematic un pointer se reprezintă ca în figura de mai jos.

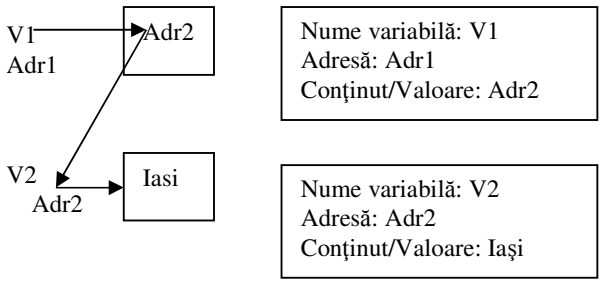


Figura 2.4. Reprezentarea unui pointer V1

Explicaţii referitoare la *pointer*:

- În limba română *pointer* înseamnă *ac indicator* sau simplu *indicator*, adică un instrument care ne indică o direcţie;
- Ştiind numele variabilei *pointer*, în cazul nostru *v1*, putem ajunge la valoarea variabilei *v2*.

2.2.3 Liste

O listă reprezintă o colecţie omogenă şi secvenţială de date. Astfel, aceasta constituie cea mai comună dată structurată. Schematic cea mai simplă listă se reprezintă ca în figura de mai jos.

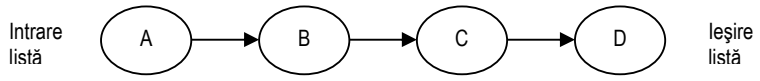


Figura 2.5. Reprezentare listă simplă

O asemenea listă mai poartă denumirea de *listă liniară* sau *simplă*, adică un set de date elementare stocate în locaţii de memorie consecutive. Într-o asemenea listă vom distinge:

- *Nodul*, adică acea componentă în care se conţine o dată elementară;
- *Lungimea*, adică numărul de noduri din listă,
- *Definiţia* pentru stabilirea ordinii nodurilor, adică pentru fiecare nod există un predecesor şi un succesor.

În situaţia în care definiţia pentru stabilirea ordinii nodurilor conţine o regulă care conduce la o succesiune între ultimul nod şi primul nod, atunci discutăm de *liste circulare*.

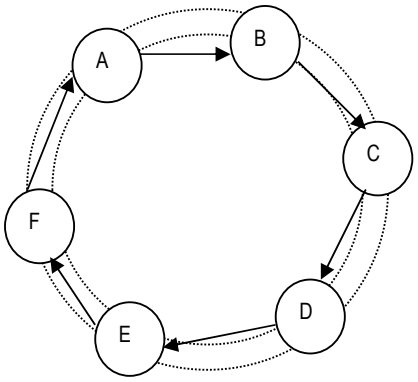


Figura 2.6. Reprezentare listă circulară

Unei liste i se pot aplica operaţii de actualizare (modificare, inserare şi ştergere) noduri din listă, operaţii de concatenare cu altă listă şi de numărare a nodurilor declarate.

Pentru implementarea unei liste se pot folosi două strategii:

a. Implementarea secvenţială sau statică, ceea ce presupune atribuirea de locaţii succesive de memorie în corespondenţă cu ordinea nodurilor din listă.

b. Implementarea înlănţuită sau dinamică. În acest caz, fiecare nod conţine două părţi: informaţia propriu-zisă şi adresa nodului succesor.

În practică se întâlnesc două cazuri particulare de liste:

a. Stive. O *stivă* (*stack*) este o listă liniară cu proprietatea că operaţiile de inserare/ extragere a nodurilor se fac în/ din coada listei. Dacă nodurile A, B, C, D sunt inserate într-o stivă în aceasta ordine, atunci primul nod care poate fi extras este D. In mod echivalent, spunem ca ultimul nod inserat va fi şi primul şters. Din acest motiv, stivele se mai numesc şi liste *LIFO* (**L**ast **I**n **F**irst **O**ut), sau liste *pushdown*.

b. Cozi. O *coada* (*queue*) este o listă liniară în care inserările se fac doar în capul listei, iar extragerile doar din coada listei. Cozile se numesc şi liste *FIFO* (**F**irst **I**n **F**irst **O**ut).

2.2.4 Arbori

Arborii reprezintă o dată structurată cu două-dimensiuni în care nodurile formează o ierarhie. Astfel, o asemenea structură de date dispune de un *nod rădăcină* (plasat fie în vârf, fie la baza arborelui). La rândul său. un *nod rădăcină* dispune de unul sau mai multe niveluri de *noduri-copil*, care pot să aibă, la rândul lor, proprii lor *noduri-copil*. Astfel, putem spune că într-un arbore deosebim: *nodul rădăcină*, *noduri copil* şi *noduri-părinte*. Mai multe noduri subordonate succesiv unul altuia formează o *ramură*, iar un nod care nu are subordonat un nod-copil se numeşte *nod frunză*. În forma sa cea mai sintetică un arbore se reprezintă ca în figura 2.7.

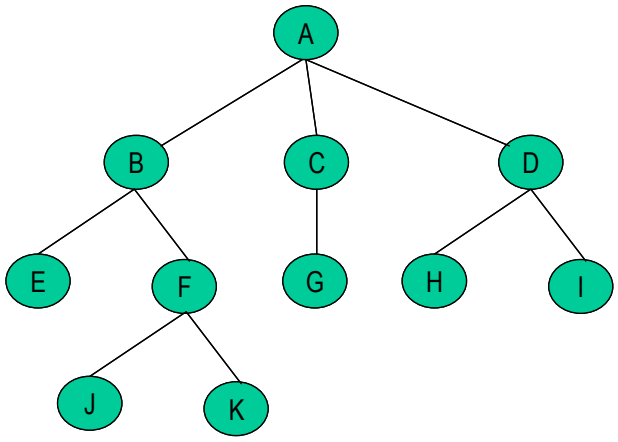
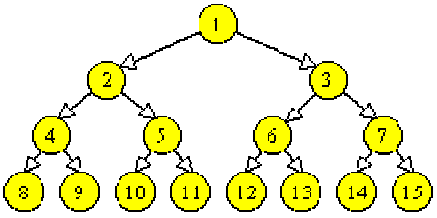


Figura 2.7 Exemplu de structură ierarhică arborescentă

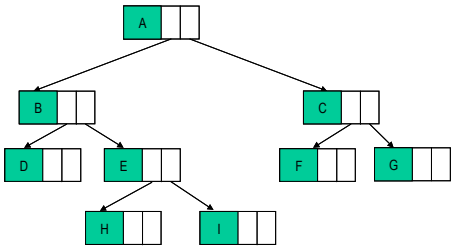
Pe exemplul din figura precedentă particularizăm următoarele tipuri de noduri:

- nod rădăcină: a;
- noduri copil: b, c, d, e, f, g, h, i, j şi k;
- nod părinte: b faţă de e şi f; c faţă de g; d faţă de h şi i; f faţă de j şi k;
- ramură: a, b şi e;
- noduri frunză: e, j, k, g, h şi i;

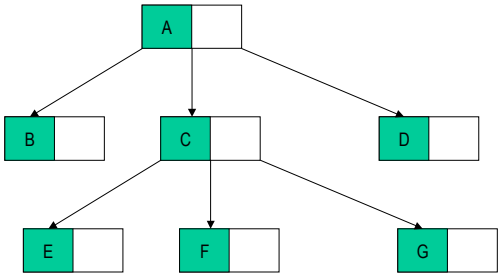
În literatura de specialitate sunt prezentaţi: arbori cu rădăcină (figura 2.8.a), arbori binari (figura 2.8.b), arborii multi-cale (figura 2.8.c).



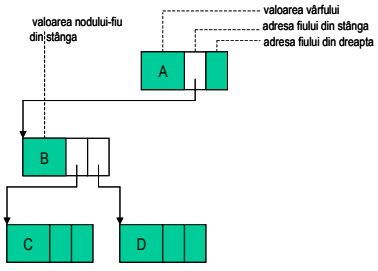
a. Arbore cu rădăcina



b. Arbore binar



c. Arbori multi-cale



d. Memorarea valorilor şi adreselor în noduri

Figura 2.8 Reprezentări grafice ale arborilor

Reprezentarea *unui arbore cu rădăcină* se poate face prin adrese cu ajutorul listelor înlănţuite. În cazul *arborelui binar* se observă că fiecare nod-părinte are în subordine câte

două noduri-copil. După cum se observă şi din figura 2.8.d., fiecare nod va fi memorat în trei locaţii diferite ale unei listei, după cum urmează: o locaţie va fi folosită pentru valoarea propriu-zisă a nodului, o a doua locaţie va fi destinată pentru memorarea adresei celui mai vârstnic fiu, iar a treia locaţie pentru adresa următorului frate. Cu alte cuvinte, legătura între un nod-părinte şi un nod-copil se realizează prin intermediul *pointerilor*.

Observaţie: Sesizăm faptul că pentru *arbori* se face trimitere la *liste*, pentru a căror reprezentare se folosesc *tablourile bidimensionale*. Iată de ce tablourile reprezintă cea mai simplă dată structurată.

2.2.5 Structuri de date externe: articolul şi fişierul

Fişierul reprezintă o colecţie ordonată de date. Urmare a faptului că stocarea datelor se face pe suporturi de memorie externă, fişierul este definit şi ca structură de date externă.

Structura ordonată de date din cadrul fişierului poartă denumirea de *articol*, care la rândul său este constituit dintr-o mulţime ordonată de valori ale căror caracteristici aparţin unei entităţi din domeniul unei probleme. Caracteristicile descrise prin intermediul valorilor din articolele unui fişier le vom regăsi sub denumirea de *câmpuri*.

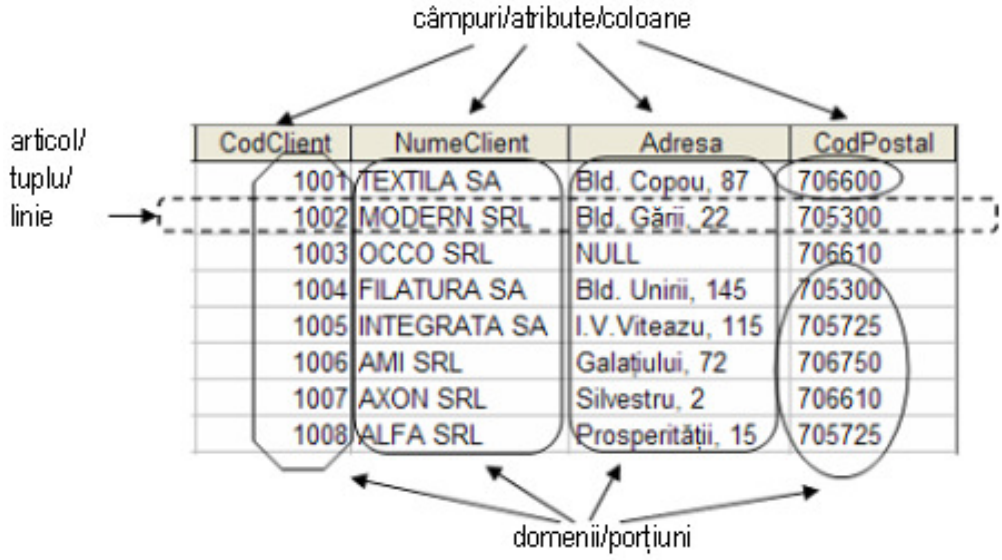


Figura 2.9 Reprezentări ale structurilor de date externe de tip articol şi fişier

Capitolele 4, 5 şi 6 prezintă exemple suplimentare de structuri de date de tip fişiere, articole şi tabele.

Din punct de vedere al activităţilor de programare este interesant să cunoaştem metoda de organizare şi tipul de acces aferente structurilor de date externe.

a. Metoda de organizare poate fi una din următoarele: secvenţială, relativă şi indexată;
b. Tipul de acces la date în operaţii de citire şi scriere a valorilor din /în câmpuri. Se poate folosi accesul secvenţial sau accesul direct.

2.3 STRUCTURI ALGORITMICE DE PRELUCRARE

Principala sarcină a unui mediu de programare (instrument software) constă în a permite *definirea structurilor de date* într-un mod eficient şi de a *asigura accesul, manipularea, actualizarea şi controlul* asupra acestora prin intermediul algoritmilor şi structurilor de prelucrare /control. Structurile de prelucrare /control constituie aportul cel mai de seamă al metodei de programare structurată, care a impus utilizarea a trei tipuri de structuri fundamentale, toate cu o singură intrare şi o singură ieşire. Cele trei tipuri de structuri fundamentale sunt:

- secvenţiale;
- alternative;
- repetitive.

În figura de mai jos punem în evidenţă aceste tipuri şi instrucţiunile corespunzătoare.

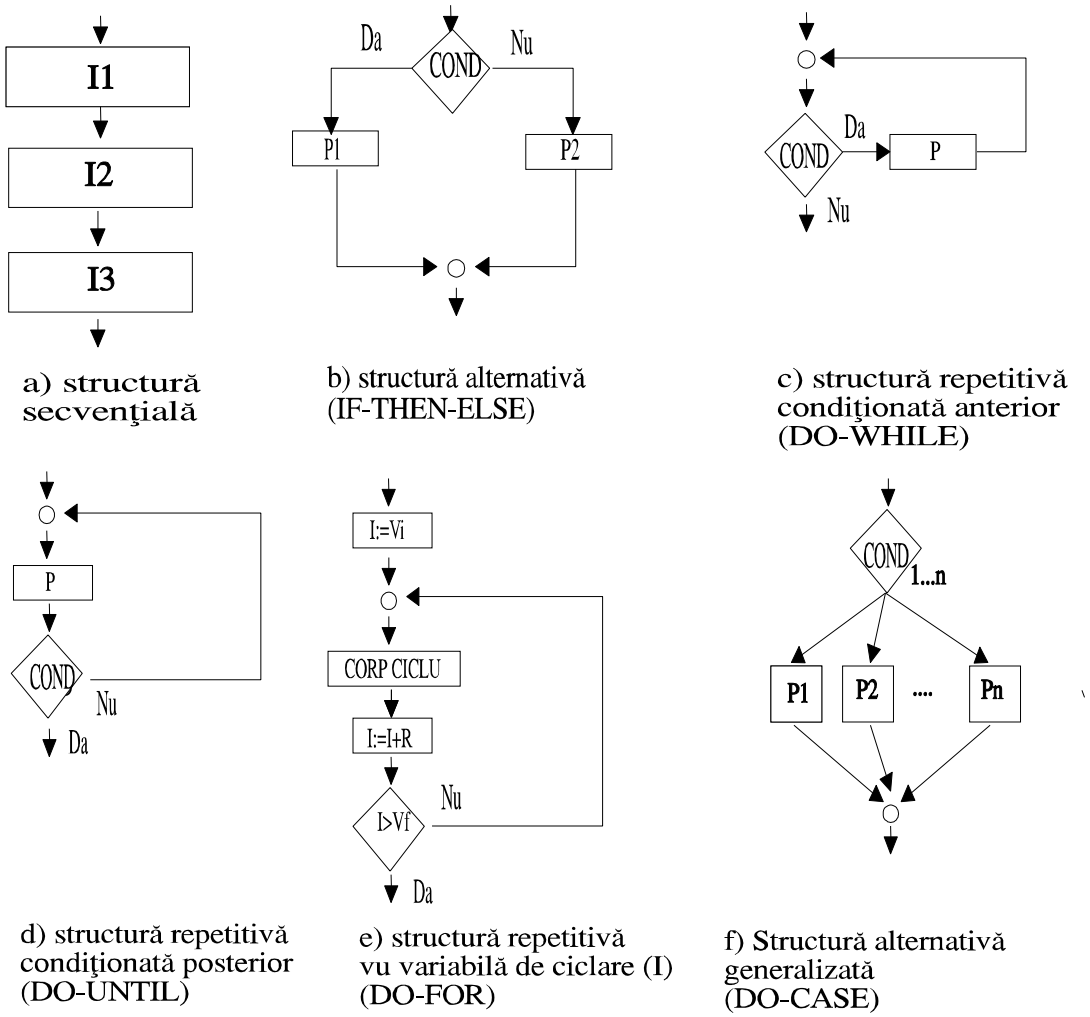


Figura 2.10 Structurile de control fundamentale

În Visual *Basic for Application* (VBA) cele trei structuri de control fundamentale se codifică în mod direct prin comenzi specifice acestui mediu de programare.

A. Structuri secvenţiale

Ca în orice mediu de programare VBA asigură operaţii de atribuire şi operaţii de intrare/ ieşire, după cum urmează:

- *Atribuirea* se realizează cu instrucţiune LET sau direct cu folosirea semnului =, ca în exemplul precedent.

```
Sub atribuire()
        ‘ precizare varibilele care se folosesc mai jos trebuie in principiu declarate (create) inainte
        LET cont=212                        ‘ se atribuie valoarea numerica 212 pentru variabila cont
       Den = “Mijloace fixe” ‘ se atribuie şirul „Mijloace fixe” pentru variabila den
        Tip_C= True                         ‘ valoarea True este atribuită conturilor de activ
        Data=#12/31/2004#                   ‘ se atribuie valoarea 31.12.2004 variabilei data
        ‘ Nota: Trebuie se foloseste semnul #, iar formatul este ll/zz/aaaa
End  Sub
```

- *Operaţia de atribuire* prin citirea datelor introduse de utilizator este asigurată de instrucţiunea **InputBox**. Cu alte cuvinte, această instrucţiune ne permite introducerea de la tastatură a valorilor pentru o anumită variabilă. Exemplul de mai sus se transformă în următorul:

```
Sub citire()
        Cont = InputBox(“Simbol cont”,”Introduceti o valoare”, 0)    ‘se permite citirea (preluarea) în
                                                                        ‘variabila cont a raspunsului dat de
                                                                        ‘ utilizator
        Den = InputBox(“Denumire cont:”,”Introduceti o valoare”, “”)
        Data= InputBox(“Data operatiei:”,”Introduceti o valoare in formatul ll/zz/aaaa”, 0)
End  Sub
```

Formatul instrucţiunii InputBox este următorul:

InputBox(mesaj, [titlu, valoare_implicita])

Precizări:

- *Mesaj* trebuie scris între ghilimele sau se poate prelua dintr-o variabilă de tip şir. Este textul ce apare ca explicaţie în interiorul ferestrei;
- Titlu şi valoare_implicită sunt facultative;
- Titlu trebuie scris intre ghilimele sau se poate prelua dintr-o variabilă de tip şir. Este textul ce apare pe linia de titlu a ferestrei;
- Valoare_implicită este valoarea care va apare în rubrica de completat;
- Fereastra rezultată are şi butoanele predefinite *OK* şi *Cancel* (vezi figura 2.11)

Spre exemplu, din procedura *Citire*() rezultatul executării liniei

Cont = InputBox(“Simbol cont”,”Introduceti o valoare”, 0)

este prezentat în figura 2.17.

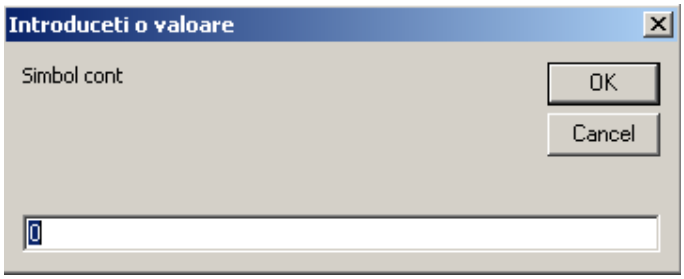


Figura 2.11 Rezultatul execuţiei liniei Cont = InputBox("Simbol cont","Introduceti o valoare", 0)

- *Operaţia de ieşire* este uşor de evidenţiat cu ajutorul instrucţiunii MsgBox, care permite afişarea unui text într-o fereastră de dialog. Cea mai simplă utilizare este atunci când dorim sa vizualizăm conţinutul unei variabile, ca în exemplul de mai jos.

```
Sub citire()
        Cont = InputBox("Simbol cont","Introduceti o valoare", 0)
        Den = InputBox("Denumire cont:","Introduceti o valoare", "")
        Data= InputBox("Data operatiei:","Introduceti o valoare in formatul ll/zz/aaaa", 0)

        MsgBox ("Informatii despre cont")              ' afiseaza text simplu
        MsgBox ("Denumire cont: " & Den)               ' afiseaza denumire cont cu text in fata
        MsgBox ("Simbol cont: " & Str(cont))           ' afiseaza textul simbol cont: dupa care simbolul
introdus
        MsgBox ("" & data)                             ' afiseaza o data calendaristica

End  Sub
```

Prezentăm în figura 2.12 fereastra afişată pentru instrucţiunea MsgBox ("Denumire cont: " & Den) şi a instrucţiunii *MsgBox* ("Denumire cont: " & Den).

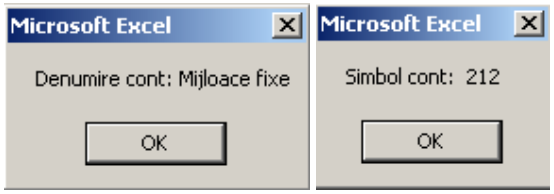


Figura 2.12 Rezultate la execuţia instrucţiunilor de tip MsgBox

În chenarul de mai jos oferim spre exemplificare o utilizare mai complicată a instrucţiunii *MsgBox* pentru preluarea răspunsului de la utilizator.

```
Sub MesajBox()
        Dim Mesaj, Butoane, Titlu, Help, Raspuns, Rasp      ' Definim variabilele de lucru
        Mesaj = "Doriti sa continuati ?"                    ' Definim mesajul
        Butoane = vbYesNo + vbCritical + vbDefaultButton2   ' Definim buttoanele
        Titlu = "Titlu fereastra"                           ' Definim titlul ferestrei
        Help = "Ajutor.txt"                                 ' Definim fisierul de ajutor
        Raspuns = MsgBox(Mesaj, Butoane, Titlu, Help, 1000)
        If Raspuns = vbYes Then                             ' Selectie Yes
            Rasp = "Yes"                                    ' Executa o actiune pentru Yes
        Else                                                ' Selectie No
            Rasp = "No"                                     ' Executa o actiune pentru No
        End If
End Sub
```

În figura 2.13. prezentăm rezultatul după lansarea în execuţie a procedurii de mai sus.

Figura 2.13 Preluare răspuns prin MsgBox

B. Structuri de control alternative

Aşa cum rezultă şi din figura 2.10 în cadrul structurilor de control alternative distingem:

- structuri de control alternative duble;
- structuri de control alternative cu o ramură vidă;
- structuri de control alternative generalizate (multiple).

a. Codificarea structurilor de control alternative duble şi a celor cu ramură vidă se realizează cu ajutorul comenzii: **IF … THEN … ELSE … ENDIF**. Formatul acestei comenzi este:

```
IF <condiţie> THEN
        <set_comenzi1>
[ELSE
        <set_comenzi2>]
END IF
```

Pentru codificarea structurii de control alternative cu o ramură vidă formatul comenzii IF devine:

```
IF <condiţie> THEN
    <set_comenzi>
END IF
```

Se observă că în formatul precedent ramura ELSE este opţională fiind inclusă în paranteze drepte.

Exemplu de utilizare – Vom exemplifica această structură de control pe selecţia folder-ului (directorului) de lucru pentru o aplicaţie ce poate fi lansată în laboratoarele facultăţii sau pe un calculator personal (acasă sau la serviciu).

```
Sub Mod_IF()
        ' construim fereastra de interogare a dorintei utilizatorului
        Dim dir As Object
        Set dir = CreateObject("Scripting.FileSystemObject")
        Dim cale, cale_nou
        Dim Mesaj, Butoane, Titlu, Help, Raspuns                    'Definim variabilele de
lucru
        Mesaj = "Lucrati in laboratoarele FEAA <Yes-Da; No-Nu>?"     'Definim mesajul
        Butoane = vbYesNo + vbCritical + vbDefaultButton2           'Definim buttoanele
```

```
        Titlu = "Selectati un raspuns !"                                    'Definim titlul ferestrei
        Help = "Ajutor.txt"                                                 'Definim fisierul de ajutor
        Raspuns = MsgBox(Mesaj, Butoane, Titlu, Help, 1000)                 'Stocarea raspunsului in
Raspuns

        ' **** Selectia unitatii de lucru D sau Z

          If Raspuns = vbYes Then
                cale = "Z:"
          Else
                cale = "D:"
          End If
          ChDir (cale)

        ' **** Testarea existentei directorului VBA pe unitatea de disc selectata
              cale_nou = "" & cale + "\VBA"          'construirea caii de lucru complete

        If dir.FolderExists(cale_nou) Then                  'testarea existentei directorului VBA
           MsgBox "VBA este directorul de lucru de pe " & cale
        Else
           Raspuns = MsgBox("Nu exista directorul VBA pe " & cale, vbOKOnly, "Informatie")
           MkDir (cale_nou)                                 'crearea noului director VBA in cazul in care
nu exista
           Raspuns = MsgBox("VBA a fost creat si va fi directorul de lucru de pe " & cale, vbOKOnly,
"Informatie")
        End If
        ChDir (cale_nou)                                    'stabilirea directorului de lucru

End Sub
```

În procedura de mai sus se observă două instrucţiuni IF una după alta. Există însă posibilitatea ca pe ramura *Else* a unei structuri IF să nu fie nici o secvenţă, ceea ce ne conduce la o structură alternativă cu o ramură vidă de forma IF … THEN … END IF, dar şi la posibilitatea ca pe oricare din cele două ramuri să apară o altă structură alternativă, ceea ce ne-ar conduce la un IF imbricat de forma **IF … THEN** *If …Then … Else … End If* **ELSE … END IF**.

b. Codificarea structurilor de control alternative generalizate se realizează cu ajutorul comenzii: **SELECT CASE … CASE … CASE ELSE … END SELECT**. Formatul general pentru această comandă este următorul:

```
SELECT CASE <var>
    CASE <var_1>
            <set_comenzi_1>
    [CASE <var_2>
            <set_comenzi_2>]
    [CASE <var_k>
            <set_comenzi_k>]
            […]
    [CASE <var_m>
            <set_comenzi_m>]
    [CASE ELSE
            <set_comenzi>]
END SELECT
```

Modul de utilizare a acestei comenzi este simplu şi constă în a identifica cazurile de descris, care se vor reprezenta sub forma unor condiţii. De fiecare dată se compară conţinutul variabilei **<var>** cu conţinutul variabilelor **<var_1>**, **<var_2>** … . Dacă se ajunge să fie indeplinită o condiţie <*var*>=<*var_k*> se va executa *set_comenzi_k* (*k poate fi o valoare din intervalul 1,m*). În cazul în care nu se îndeplineşte nici una din condiţii, se transferă contolul pe ramura lui **CASE ELSE** spre a se executa setul de comenzi <*set_comenzi*>.

Exemplu de utilizare – Dorim să aflăm ziua din cadrul săptămânii pornind de la data naşterii. Se ştie că datele calendaristice sunt reprezentate ca valori numerice, iar restul împărţirii la 7, la care se adaugă 1, ne asigură obţinerea unui număr în intervalul 1-7.

```
Function zi(numzi As Byte) As String
    Select Case numzi
        Case 1
            zi = "Luni"
        Case 2
            zi = "Marti"
        Case 3
            zi = "Miercuri"
        Case 4
            zi = "Joi"
        Case 5
            zi = "Vineri"
        Case 6
            zi = "Sambata"
        Case 7
        zi = "Duminica"
    End Select
End Function

Sub case_zi()
        Dim varzi As Date
        varzi = InputBox("Introduceti data Dvs de nastere in format ll/zz/aaaa")
        MsgBox ("Sunteti nascut(a) intr-o zi de " + zi((Int(varzi) Mod 7) + 1))
End Sub
```

Observaţie: *(Int(varzi) Mod 7) + 1* asigură obţinerea restului la împărţirea lui varzi la 7 in intervalul 1 – 7.

Exemplu de utilizare – În funcţie de distanţa în km la care se află localitatea dvs. se va afişa un mesaj de apreciere a apropierii faţă de Iaşi.

Sub dist()

```
    Dim d
    d = InputBox("Introduceti distanta pana la localitatea dvs. in Km.")
    Select Case d
    Case Is <= 50
       MsgBox ("Localitatea este foarte aproape, la distanta de " + Str(d) + " km.")
    Case 51 To 100
       MsgBox ("Localitatea este aproape, la distanta de " + Str(d) + " km.")
    Case 101 To 250
       MsgBox ("Localitatea este departe, la distanta de " + Str(d) + " km.")
    Case Is > 250
```

```
        MsgBox ("Localitatea este foarte departe, la distanta de " + Str(d) + " km.")
    Case Else
        MsgBox ("Distanta introdusa trebuie sa fie un numar intreg!")
    End Select

End Sub
```

C. Structuri de control repetitive

În VBA structurile de control repetitive sunt realizate astfel:

- **WHILE … WEND**, pentru structuri repetitive condiţionate anterior. Setul de comenzi se execută *cât timp condiţia este adevărată*.
- **FOR … NEXT**, pentru structuri repetitive cu număr stabilit de paşi.

WHILE … WEND

În VBA această structură repetitivă este considerată cea mai simplă, deoarece nu oferă posibilitatea părăsirii forţate a setului de comenzi cu o instrucţiune de tipul **EXIT**. Formatul său este:

WHILE conditie	**CÂT TIMP conditie**
<set_comenzi>	**<set_comenzi>**
WEND	**SFÂRŞIT_CÂT_TIMP**

Iată un exemplu de utilizare a acestei structuri de prelucrare pe exemplul introducerii de date în tabloul cu două dimensiuni pentru produsele care au intrat în magazin într-o perioadă de timp stabilită de utilizator.

```
Sub Wend_tablou_2()
Dim i  As Integer   ' definirea variabilei pentru control executie
Dim var As Integer  ' definirea variabilei de ciclare
i = InputBox("Numar de intrari in cadrul zilei: ")
Dim aprov(5, 100)   ' definirea tabloului
var = 0
   While var < i
        aprov(0, var) = InputBox("Denumire produs: " & (var + 1))
        aprov(1, var) = InputBox("Pret de intrare: " & (var + 1))
        aprov(2, var) = InputBox("Unitate de masura: " & (var + 1))
        aprov(3, var) = InputBox("Cantitate:" & (var + 1))
        aprov(4, var) = InputBox("Data intrarii (ll/zz/aaaa):")
      var = var + 1
   Wend
End Sub
```

Observaţie: Pentru a nu se repeta la infinit este necesară introducerea unei clauze de actualizare a valorii variabilei de ciclare var (*vezi linia var = var + 1*).

FOR … NEXT

Această structură oferă posibilitatea prestabilirii numărului de execuţii pentru set_comenzi. Din punct de vedere logic se aseamănă cu WHILE … WEND, cu precizarea că actualizarea lui i se face cu clauza NEXT . Formatul său este:

FOR i =1 to n	**PENTRU i = 1 la N**
<set_comenzi>	**<set_comenzi>**
NEXT i	**URMĂTORUL i**

Vom adapta exemplul precedent în cazul utilizării structurii de prelucrare FOR … NEXT, dar pe zile ale săptămânii.

```
Sub For_tablou_2()
        Dim i  As Integer   ' definirea variabilei de ciclare
        Dim aprov(5, 100)   ' definirea tabloului
        Dim zi As String
        i = 0
          For i = 0 To 6
            Select Case i
              Case 0
               zi = "Luni"
              Case 1
               zi = "Marti"
              Case 2
               zi = "Miercuri"
              Case 3
               zi = "Joi"
              Case 4
               zi = "Vineri"
             Case 5
              zi = "Sambata"
             Case 6
              zi = "Duminica"
           End Select
            aprov(0, i) = InputBox("Denumire produs intrat " & zi & ":")
            aprov(1, i) = InputBox("Pret de intrare:")
            aprov(2, i) = InputBox("Unitate de masura:")
            aprov(3, i) = InputBox("Cantitate:")
            aprov(4, i) = InputBox("Data intrarii (ll/zz/aaaa):")
          Next i
End Sub
```

DO … LOOP

Această structură de prelucrare de tip repetitiv permite utilizarea a două clauze diferite: pentru transferul controlului în cazul îndeplinirii condiţiei (WHILE) şi pentru transferul controlului în cazul neîndeplinirii condiţiei (UNTIL). În varianta cu WHILE DO … LOOP corespunde structurii de control repetitive condiţionate anterior, în timp ce în varianta cu UNTIL, permite codificarea structurilor de control condiţionate posterior (adică set_comenzi se execută înaintea verificării condiţiei, deci cel puţin o dată).

```
DO WHILE <conditie>
    <set_comenzi_1>
     [EXIT DO]
     <set_comenzi_2>
LOOP

DO UNTIL <conditie>
     <set_comenzi_1>
      [EXIT DO]
      <set_comenzi_2>
```

```
LOOP

DO <conditie>
   <set_comenzi_1>
    [EXIT DO]
    <set_comenzi_2>
LOOP [UNTIL|WHILE]
```

Oferim două exemple de utilizare a structurii repetitive DO ... LOOP

```
Sub citire()
    DO UNTIL CONT<900 and CONT>99
        Cont = InputBox("Simbol cont","Introduceti o valoare", 0)    'se permite citirea (preluarea) în
                                                                      'variabila cont a raspunsului dat de
                                                                      ' utilizator
    LOOP
        Den = InputBox("Denumire cont:","Introduceti o valoare", "")
        Data= InputBox("Data operatiei:","Introduceti o valoare in formatul ll/zz/aaaa", 0)
End  Sub
```

```
Sub citire()
    DO WHILE CONT>900 and CONT<100
        Cont = InputBox("Simbol cont","Introduceti o valoare", 0)    'se permite citirea (preluarea) în
                                                                      'variabila cont a raspunsului dat de
                                                                      ' utilizator
    LOOP
        Den = InputBox("Denumire cont:","Introduceti o valoare", "")
        Data= InputBox("Data operatiei:","Introduceti o valoare in formatul ll/zz/aaaa", 0)
End  Sub
```

Exemple de întrebări teste grilă

1. Care structură de control este denumită incorect:
   a. Secvențială
   b. Repetitivă
   c. Iterativă
   d. Repetitiv-Indexată
2. Tipul de dată Arbore este:
   a. Elementar
   b. Structurat
   c. Stivă