Bogdan PRICOP
Gheorghe BĂDĂRĂU

Cap 7. Instrucțiuni de prelucrare

- ***Fișiere. Înregistrări***
- ***Instrucțiunea READ***
- ***Instrucțiunea WRITE***
- ***Instrucțiunea de atribuire***
- ***Scheme logice***
- ***Pseudocod***
- ***Teorema de structură. Construcții de control***

Fișiere. Înregistrări

Un fișier este o resursă a calculatorului folosită pentru înregistrarea discretă a datelor într-un dispozitiv de stocare. Există diferite tipuri de fișiere concepute pentru scopuri diferite.

Un fișier poate fi proiectat pentru a stoca o imagine, un mesaj scris, un videoclip, un program de calculator sau o mare varietate de alte tipuri de date. Unele tipuri de fișiere pot stoca mai multe tipuri de informații simultan.

Prin utilizarea unor programe, o persoană poate deschide, citi, schimba și închide un fișier. Fișierele computerului pot fi redeschise, modificate și copiate de nenumărate ori.

În mod tipic, fișierele sunt organizate într-un sistem de fișiere, care urmărește unde sunt acestea și le permite oamenilor să le acceseze.

Fișiere. Înregistrări

În sistemele de operare moderne, fișierele sunt organizate într-o rețea unidimensională de octeți. Formatul unui fișier este definit de conținutul său, deoarece un fișier este doar un recipient pentru date. Pe anumite platforme, formatul este de obicei indicat de extensia fișierului, fiind specificate regulile pentru modul în care octeții trebuie să fie organizați și interpretați.

De exemplu, octeții unui fișier text simplu (.txt în Windows) sunt asociați cu caractere ASCII sau UTF-8, în timp ce octeții fișierelor imagine, video și audio sunt interpretați altfel. Cele mai multe tipuri de fișiere alocă de asemenea câteva octeți pentru **metadate**, ceea ce permite unui fișier să transmită câteva informații de bază despre el însuși.

Instrucțiunea READ

Instrucțiunea de citire (READ) are forma

read vname [,vname]...

Și este alcătuită din cuvântul read ce înseamnă "citește" și o listă de variabile.

Exemplu: Instrucțiunea - read alfa – unde lui alfa îi este atribuită valoarea 204 se execută astfel: procesorul caută în memorie celula cu adresa simbolică alfa. Presupunem că nu găsește o astfel de celulă; atunci alege o celulă de memorie liberă, o numește alfa și în ea stochează valoarea 204. În urma execuției instrucțiunii read alfa variabila alfa a fost definită, astfel în memorie va există o celulă cu adresa simbolică alfa și în acea celulă este stocată valoarea variabilei, 204.

Instrucțiunea WRITE

Instrucțiunea de scriere (WRITE) are forma

write entitate [,entitate]...

Și este alcătuită din cuvântul write ce înseamnă „scrie" și o listă de entități. Fiecare entitate poate fi o variabilă sau o expresie.

Exemplu: Instrucțiunea - write kapa - se execută astfel: procesorul caută în memorie celula kapa, o consultă (află valoarea stocată acolo fără a o distruge și fără a-i altera conținutul) și realizează în fișierul de ieșire o înregistrare cu valoarea variabilei kapa; de exemplu, dacă valoarea variabilei kapa este 5, se creează o nouă înregistrare ce conține valoarea 5.

Evident instrucțiunea write nu are sens dacă în prealabil variabilele ce apar în entitățile din lista de ieșire nu au fost definite anterior.

Instrucțiunea de atribuire

Instrucțiunea de atribuire are forma

vname ← expr

unde vname este o variabilă, expr este o expresie iar ← reprezintă semnul de atribuire. Dacă expr este o expresie numerică, variabila vname este o variabilă numerică. Dacă expr este o expresie logică, vname este o variabilă logică. Dacă expr este o expresie caracter atunci vname este o variabilă caracter. Instrucțiunea se execută astfel: cu datele existente în memorie procesorul evaluează expr şi apoi rezultatul este stocat în celula cu numele vname.

Scheme logice

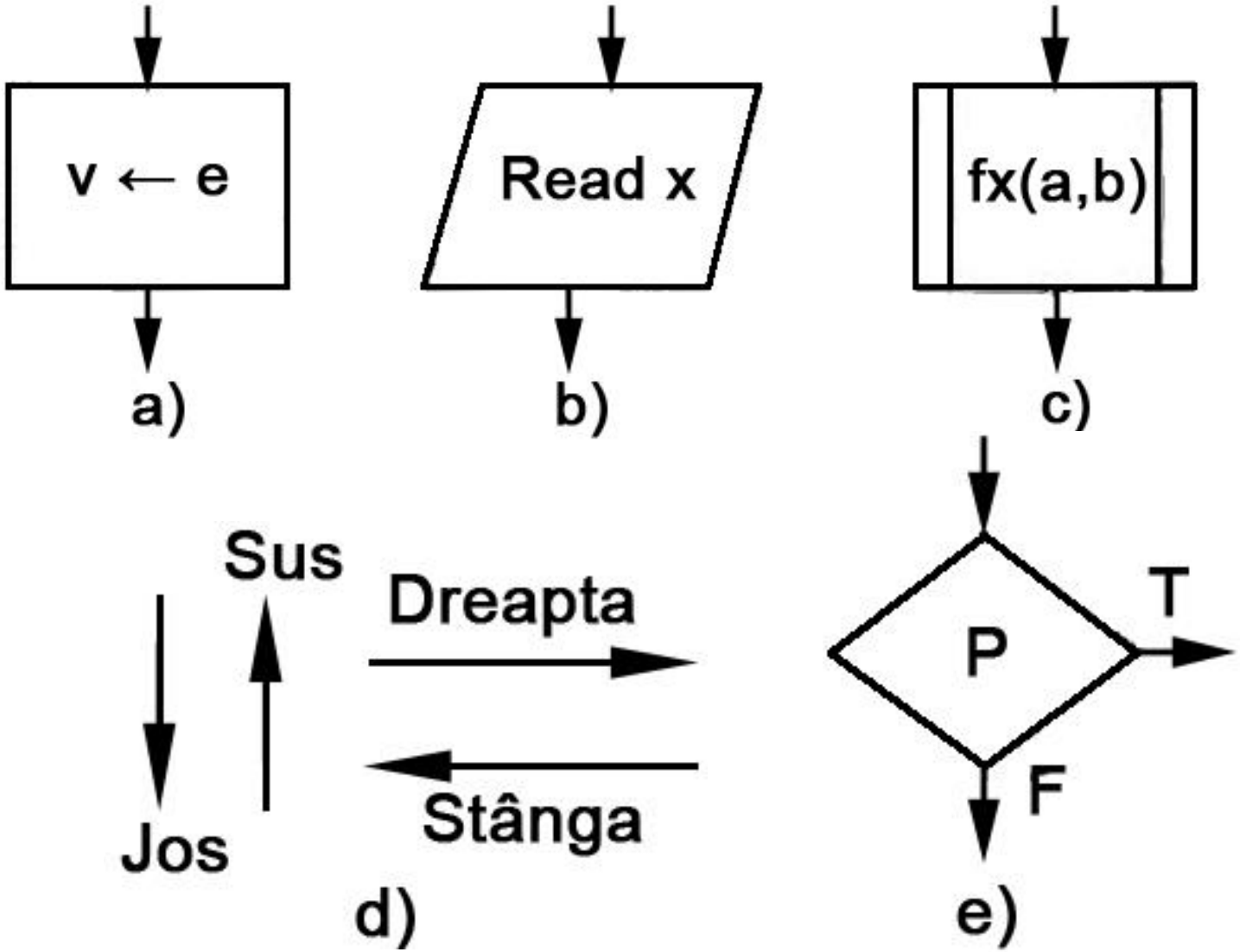
Descrierea logicii programului, adică instrucțiunile de prelucrare și ordinea lor de execuție se face de obicei cu ajutorul schemelor logice.

Schema logică este o reprezentare grafică bidimensională a algoritmului cu ajutorul unor simboluri speciale. Schema logică este mai intuitivă decât programul scris într-un limbaj de programare. De aceea erorile logice ale programului se descoperă mai repede pe schema logică.

Din punct de vedere matematic schema logică este un grafic ce poate avea noduri de trei feluri: ***nod funcțional, nod predicativ și nod de reunire***.

Nodul funcțional sau nodul de prelucrare se poate reprezenta ca în figura următoare (a-c). Acesta are o intrare și o ieșire și reprezintă o operație de prelucrare a informației.

Scheme logice



Scheme logice

Nodul predicativ are o intrare și două ieșiri (e). Predicatul P înscris în romb și reprezintă o condiție logică ce trebuie verificată. În funcție de rezultat, true (T) sau false (F) se alege una din ieșiri. **Nodul de reunire** se reprezintă printr-un cerculeț plin în care intră două linii și iese o singură linie. În nodul de reunire nu se efectuează nici o operație de prelucrare, nodul de reunire este o simplă reuniune cu două intrări şi o ieșire.

Simbolurile se pot grupa în simboluri de bază, simboluri adiționale și simboluri specializate de prelucrare.

Simbolul proces reprezintă procesul de execuție a unei anumite operații sau a unui grup de operații în urma căruia rezultă schimbarea formei sau valorii informației. Forma este dreptunghi cu raportul înălțime - lățime 2/3 (a).

Scheme logice

Simbolul intrare/ieşire reprezintă funcţia generală de input-output. Mediul suport de intrare/ieşire nu este specificat. Forma este paralelogram cu unghiul ascuțit de 75° și raportul înălțime-lăţime 2/3 (b).

Simbolul linie și vârf de săgeată arată direcția fluxului de informație în și succesiunea operațiilor. Se reprezintă prin orice segment de dreaptă ce trebuie să lege două simboluri. Direcția normală este sus-jos și dreapta-stânga (d).

Simbolul decizie se folosește pentru a reprezenta o decizie ce determină una din două alternative posibile. Forma este romb cu diagonala mare orizontală și cu raportul înălțime-lățime 2/3 (e).

Simbolul proces predefinit reprezintă una sau mai multe operații de prelucrare specificate în detaliu în altă parte (procedură, altă schemă logică). Forma este aceeași cu a simbolului proces la care se adaugă 2 linii verticale (c).

Pseudocod

În locul schemelor logice adesea se folosește pseudocodul. Pseudocodul este un limbaj asemănător limbajului natural care ne permite să descriem formal logica programului fără toate detaliile concrete ale unui limbaj de programare. Textul programului scris în pseudocod se aseamănă cu textul programului scris într-un limbaj de programare, dar este mai simplu. Pseudocodul se deosebește de limbajul de programare prin două caracteristici: i) nu are reguli sintactice formale, ii) poate exprima operații care nu sunt foarte precis definite.

Programul în pseudocod se scrie instrucțiune după instrucțiune, câte una pe un rând. Instrucțiunile le separăm prin punct și virgulă; aceasta înseamnă că instrucțiunea ce urmează se va executa după ce s-a terminat execuția instrucțiunii precedente.

Pseudocod

În afara de **linii-instrucțiuni** textul pseudocodului poate conține și **linii-comentariu**. În pseudocod liniile comentariu încep cu semnul exclamării "!". Comentariile nu sunt instrucțiuni, la execuția programului comentariile fiind ignorate. Comentariul este un text scris de programator pentru a face programul cât mai inteligibil.

Teorema de structură

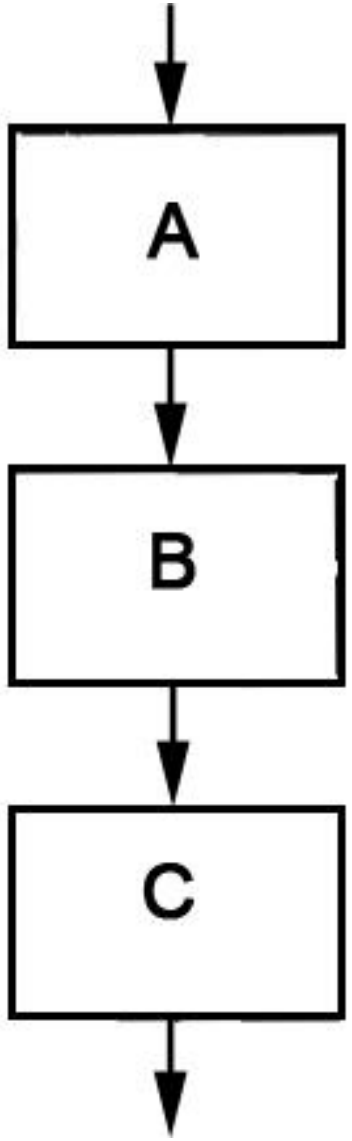
Instrucţiunile simple, citire, scriere și de atribuire, se execută în mod secvențial, una după alta, în ordinea în care sunt scrise în program. Programele conțin şi instrucțiuni de salt ce descriu o ordine de execuție diferită de cea secvențială: în funcție de îndeplinirea sau neîndeplinirea unor condiții se fac treceri în salt la instrucțiuni dinainte sau la instrucțiuni ce urmează. Prin urmare, în general, ordinea de execuție a instrucțiunilor nu coincide cu ordinea lor de scriere. În mod obișnuit fiecare programator tinde să-şi proiecteze programele folosind procedeul "încercare, eroare, corectare".

Astfel, s-a demonstrat teorema de structură potrivit căreia: *Orice program executabil este echivalent cu un program alcătuit doar cu trei tipuri de instrucțiuni simple (citire, scriere și atribuire) și trei tipuri de instrucțiuni compuse (secvența, selecția și iterația).*

Construcții de control

Instrucţiunile compuse sunt exemple de structuri de control, sau cum se mai numesc **construcții de control**.

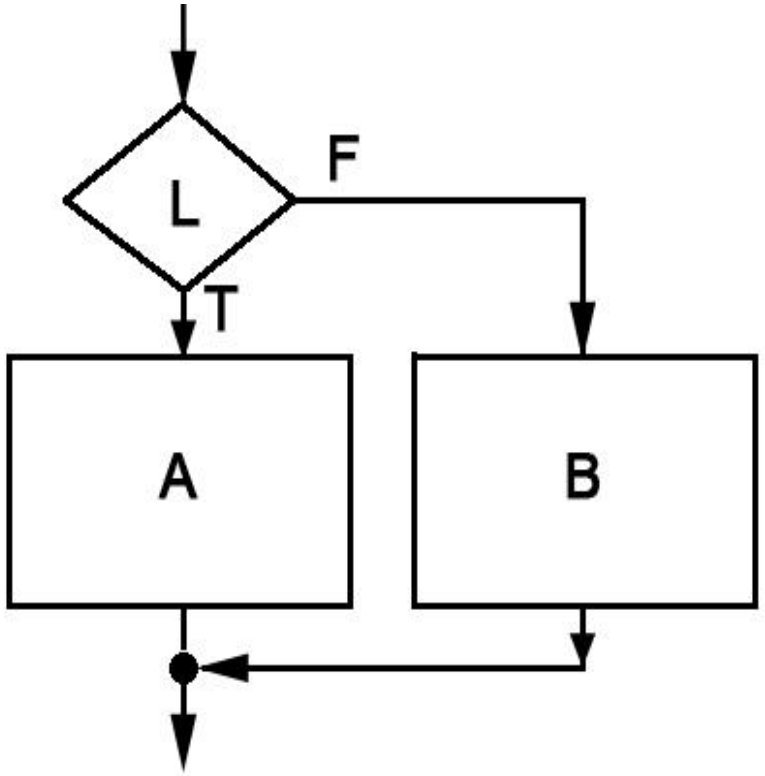
Secvența este o structură formată din două sau mai multe părți ce se execută fiecare o singură dată. Prin parte înțelegem fie o instrucțiune simplă, fie o instrucțiune compusă. Schema logică a unei secvențe este arătată mai jos. Secvența cu trei părți se execută astfel: se execută A, se execută B, se execută C și secvența ia sfârșit.



Construcții de control

Selecția este o structură de control cu două părți A și B din care se execută numai una în funcție de rezultatul unui test logic L:

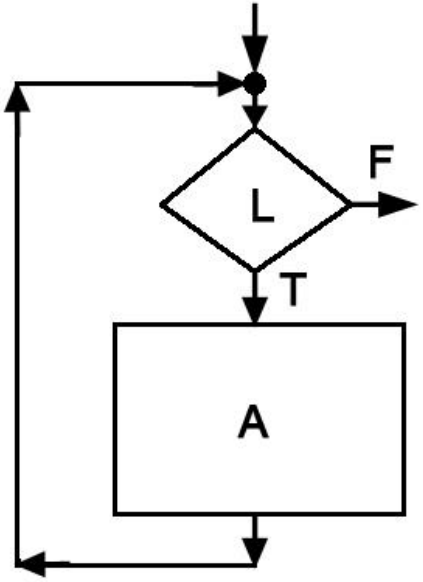
```
IF (L) THEN
        A
ELSE
        B
END IF
```



Modul de execuție al selecției este următorul: cu datele din memorie se calculează expresia logică L; dacă rezultatul este true se execută partea A și selecția ia sfârșit; dacă rezultatul este false se execută partea B şi selecția ia sfârşit.

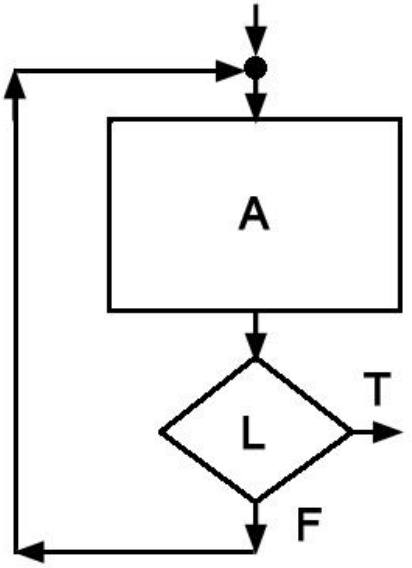
Construcții de control

Iterația este o structură de control cu o parte ce se execută de zero ori sau de mai multe ori în funcție de rezultatul unui test logic. Există două forme de iterație, iterație cu testul în capul iterației numită do while (stânga) şi iterație cu testul în coada iterației, iterație numită repeat until (dreapta).



do while (L)
A
enddo

repeat
A
until (L)



Construcții de control

Modul de execuție al iterației do while este următorul: cu datele din memorie se calculează expresia L. Dacă rezultatul este true atunci se execută A, se calculează L, dacă L este true se execută A, și așa mai departe, până când rezultatul devine false; atunci iterația ia sfârșit. Dacă la prima evaluare a expresiei logice rezultatul este false iterația ia sfârșit și partea A nu se execută niciodată. În programare iterația se mai numește buclă. Pentru ca iterația să se termine într-un număr finit de pași este necesar ca partea A să modifice valoarea unei variabile ce apare și în testul L. Dacă această condiție nu se îndeplinește avem de-a face cu o bucă infinită.

Construcții de control

Modul de execuție al iterației repeat until este următorul: se execută A, se evaluează expresia logică L, dacă rezultatul este false se execută din nou A, se evaluează L și așa mai departe până ce rezultatul devine true; atunci iterația ia sfârșit. Partea iterată A se execută în acest caz cel puțin o dată. Evident și buclele repeat until pot fi infinite, dacă la execuția după un număr finit de pași expresia logică L nu capătă valoarea true.

Pe plan mai general, programarea structurată acceptă și alte structuri de control, altele decât selecția, secvența și iterația. Un alt tip de structură de control este *Select Case* a cărei structură este prezentată în dreapta.

```
select case (e)
case (e1)
F1
case (e2)
F2
case (e3)
F3
case (en-1)
Fn-1
case default
Fn
end select
```

Construcții de control

Odată cu structura Case, deoarece rezultatul structurii nu corespunde modului natural de a gândi, trebuie să folosim instrucțiunea goto (sau go to), instrucțiune ce are sintaxa:

goto k

unde k este eticheta unei instrucțiuni. Eticheta este formată numai din cifre și se asociază unei instrucțiuni ce va fi referită de o altă instrucțiune.

Exemplu

```
10: A
    if (L) then
        B
        goto 10
    endif
```

Bibliografie

- *Octavian PETRUȘ, Fortran 90/95, Limbaj și Tehnici de programare, Editura Universității Tehnice "Gheorghe Asachi" din Iași, 2001*
- Romeo CHELARIU, Sisteme de operare şi limbaje de programare (Îndrumar de laborator), http://www.sim.tuiasi.ro/wp-content/uploads/Chelariu-indrumar-solp.pdf, 2004
- https://ro.wikipedia.org