Tipul înregistrare

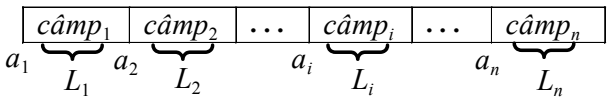
Capitolul

16

- **Declararea unui tip înregistrare**
- **Operații**
- **Implementări sugerate**
- **Probleme propuse**
- **Soluțiile problemelor**

Articolul (înregistrarea) reprezintă reuniunea unui număr fix (sau variabil) de componente numite *câmpuri*, (de regulă având tipuri diferite) care constituie logic o unitate de prelucrare. Tipurile diferite ale câmpurilor din componența sa dau articolului caracterul de structură *neomogenă*. Dacă L_i este lungimea câmpului i (i = 1, 2, ..., n), atunci valoarea adresei câmpului i este: $a_i = a_1 + L_1 + L_2 + ... + L_{i-1}$.

Caracterul neomogen al articolului, precum şi lipsa unei relații liniare între numărul de ordine al câmpului şi adresa acestuia fac ca regăsirea unui câmp să se realizeze cu ajutorul unui identificator asociat câmpului respectiv la definirea articolului.



Tipul articol (înregistrare) este o structură statică de date flexibilă, datorită faptului că, spre deosebire de tipul tablou, cuprinde un număr fix sau variabil de componente care pot fi de tipuri diferite. *Componentele* nu se indexează printr-o expresie (ca în cazul tablourilor), ci *se selectează prin intermediul identificatorilor de câmp* care se definesc la declararea tipului.

16.1. Declararea unui tip articol în limbajul Pascal

Forma generală a definiției unui ***tip înregistrare fixă*** este:

```
type identificator_tip=record
                         nume_câmp1:tip1;
                         nume_câmp2:tip2;
                         ...
                         nume_câmpn:tipn
                       end;
```

Exemplu

```
type elev=record
            nume:string;
            medie:Real;
            vârstă:0..20
          end;
var a:elev;
```

Tipul unui identificator de câmp poate fi la rândul lui tot articol.

Exemplu

```
type persoana=record
                adresa:record
                         strada:string;
                         numar:Byte
                       end;
                bi,tel:string
              end;
var a:persoana;
```

Un identificator de câmp trebuie să fie unic în cadrul propriului tip *înregistrare*.

Dacă se doreşte includerea în structura articolului a unor informații care depind de o altă informație deja prezentă în structură se formează cea de-a doua categorie, şi anume date de tip ***înregistrare cu variante***.

Declararea unui ***tip înregistrare cu variante în limbajul Pascal:***

```
type identificator_tip=record
                         nume_câmp1:tip1;
                         nume_câmp2:tip2;
                         ...
                         nume_câmpn:tipn
                         case câmp_selector: tip_selector of
                            const1:(câmpv1_1:tipv1_1; câmpv1_2:tip1_2; ...);
                            const2:(câmpv2_1:tipv2_1; câmpv2_2:tip2_2; ...);
                            ...
                       end;
```

Exemplu

```
type studii=(elem,medii,sup);                          { tip enumerare }
     personal=record
                nume:string;
```

```
                case preg:studii of
                  elem:(capac:Real);
                  medii:(bac:Real);
                  sup:(facu:string; media:Real)
                end;
var a:personal;
```

16.2. Operații

16.2.1. Atribuirea

Unei variabile de tip înregistrare i se atribuie valoarea unei alte variabile de același tip *înregistrare*. Fiind date două variabile de tip înregistrare *a* şi *b*, care au același tip, se poate realiza operația de atribuire $a \leftarrow b$ ceea ce înseamnă că valorile câmpurilor lui *a* se substituie cu valorile din *b*.

16.2.2. Selectarea unui câmp

La un câmp al unei variabile de tip înregistrare ne referim prin numele variabilei, urmat de punct, apoi numele câmpului.

Exemplu

```
a.bi, a.tel, a.adresa.numar
```

O astfel de scriere este greoaie şi pentru evitarea specificării identificatorului de tip înregistrare în fața fiecărui câmp care intervine în prelucrare, în Pascal se poate folosi instrucțiunea **with**, care simplifică modul de scriere a selectării componentelor.

Exemplu

```
with a do
  Write(adresa.strada,' ',bi,' ',tel);
```

În cazul înregistrărilor cu variante, pentru declarațiile din exemplul prezentat se pot face următoarele atribuiri de date pentru câmpurile variabilei *a*:

```
a.nume:='pop';
a.preg:=sup;
a.facu:='informatică';
a.media:=10.00;
```

sau

```
a.preg:=elem; a.capac:=9.80
```

La un moment dat este activă o singură variantă.

16.3. Implementări sugerate

Pentru a vă familiariza cu prelucrarea datelor de tip înregistrare se recomandă efectuarea următoarelor exerciții:

1. crearea unui catalog pentru o clasă;
2. afişarea în ordine descrescătoare după medie a elevilor admişi şi alfabetică a elevilor respinşi la un examen;
3. afişarea tuturor elevilor care au domiciliul într-un anumit oraş;
4. ordonarea elevilor în funcție de înălțime şi, pentru înălțimi egale, în ordinea greutății;
5. crearea unui fişier care să conțină articole privind cărțile dintr-o bibliotecă;
6. crearea unui fişier care să conțină articole privind cărțile dintr-o bibliotecă, astfel încât în cazul manualelor să se specifice disciplina, clasa pentru care este valabil manualul, autorii etc., iar pentru cărțile care nu sunt manuale, numărul autorilor, autorii, titlul, editura etc. (se va lucra cu înregistrări cu variante)
7. crearea unui fişier cu articole privind rezultatele participanților la olimpiada internațională de informatică.

16.4. Probleme propuse

16.4.1. Agendă

Un elev a creat o agendă de telefon în care a introdus numele şi numărul de telefon ale cunoştințelor sale. De asemenea, elevul doreşte să poată efectua asupra agendei sale câteva operații de forma:

- consultarea agendei cu afişarea întregului conținut;
- introducerea unei noi persoane în agendă;
- ştergerea unei persoane;
- căutare după nume;
- căutare pe baza cunoaşterii numărului de telefon.

Scrieți un program care să realizeze operațiile cerute de elev.

Date de intrare

Fişierul de intrare **`AGENDA.IN`** conține pe fiecare linie datele unei cunoştințe în forma: *nume număr de telefon*.

Actualizarea agendei se va face pe baza unor date introduse de la tastatură.

Date de ieşire

Nu vom avea un fişier de ieşire complet nou. Fişierul **`AGENDA.OUT`** va avea conținutul fişierului de intrare şi înregistrările noi, conform actualizărilor efectuate.

Restricţii şi precizări

- numărul de înregistrări este cel mult 100;
- numele este format din cel mult 25 de litere printre care nu există nici un spaţiu;
- fiecare număr de telefon începe cu cifra 0;
- la fiecare actualizare se va substitui fişierul de intrare cu cel de ieşire;
- programul va conţine un meniu din care utilizatorul va putea selecta operaţia pe care doreşte s-o efectueze;
- persoanele sunt introduse în agendă în ordine alfabetică, operaţiile de introducere a unei persoane noi şi ştergerea unei persoane nu trebuie să modifice ordonarea;
- căutarea unei persoane trebuie să se poată realiza, la cerere, după nume sau după număr de telefon;
- dacă numele unei persoane se găseşte în agendă în timpul operaţiei de introducere, se va oferi posibilitatea modificării numărului de telefon; în caz că nu se doreşte modificarea, se va introduce numărul 0, altfel noul număr.

Exemplu meniu

```
1 - consultare agenda
2 - introducerea unei noi persoane
3 - stergerea unei persoane
4 - cautare dupa nume
5 - cautare dupa telefon
6 - iesire
Optiunea ta este:
```

16.4.2. Expoziţie

Organizatorii unei expoziţii de pictură doresc să aibă o evidenţă a fiecărui tablou şi a artiştilor care au creat aceste tablouri. Organizatorii au adunat datele despre tablouri şi le-au scris într-un fişier. Pentru fiecare tablou se cunoaşte numele artistului, titlul tabloului, anul în care a fost realizat, preţul de vânzare, precum şi premiile obţinute la expoziţii locale, naţionale şi internaţionale.

Expoziţiile se codifică prin caracterele *L* (locale), *N* (naţionale) şi *E* (europene). Referitor la fiecare premiu există informaţii, după cum urmează:

- anul şi premiul în cazul expoziţiilor locale;
- anul, premiul şi localitatea în cazul expoziţiilor naţionale;
- anul, premiul, oraşul şi ţara în cazul expoziţiilor internaţionale.

Având la dispoziţie aceste date, organizatorii îşi propun să realizeze un program care să creeze următoarele liste:

1. lista lucrărilor premiate pe categorii în ordinea: locale, naţionale şi europene;
2. lista tablourilor în ordine crescătoare a preţurilor de vânzare;
3. lista artiştilor care au obţinut premiul 1 la cel puţin un concurs, indiferent de tipul acestuia.

Date de intrare
Fişierul de intrare **`EXPO.IN`** conţine atâtea linii, câte tablouri s-au expus. Pe fiecare linie este descris câte un tablou. Referitor la un tablou, pe o linie în fişier, avem următoarele date, despărţite prin câte un spaţiu:

- *nume_artist*
- *titlu_tablou*
- *an*
- *preţ*
- *nr_premii*
- *tip_expoziţie*
- *an*
- *premiu*

Dacă un tablou a fost premiat de mai multe ori, celelalte premii sunt descrise în continuare pe aceeaşi linie.

Date de ieşire
Se vor crea trei fişiere de ieşire: **`LISTA1.OUT`**, **`LISTA2.OUT`** şi **`LISTA3.OUT`**. În toate cele trei fişiere o linie conţine date referitoare la un singur tablou. În fişiere specificatorul *tip_expoziţie* va fi notat cu `L` (Locală), `N` (Naţională) sau `E` (Europeană).

Fişierul de ieşire **`LISTA1.OUT`** conţine lista lucrărilor premiate. Structura unei linii este următoarea:
tip_expoziţie **`:`** *nume_artist titlu_tablou tip_expoziţie caracteristici*
unde *caracteristici* înseamnă datele specifice fiecărui tip de expoziţie descrise în enunţ. Acestea vor fi separate printr-un spaţiu.

Fişierul **`LISTA2.OUT`** conţine tablourile în ordine crescătoare a preţurilor. Fiecare linie are forma:
nume_artist titlu_tablou preţ

Fişierul **`LISTA3.OUT`** va conţine date despre artişti şi are forma:
nume_artist titlu_tablou tip_expoziţie
În fişier artiştii vor fi ordonaţi alfabetic.

Restricţii şi precizări
- numărul de tablouri este cel mult 100;
- datele de tip **`string`** au cel mult 20 de caractere.

Exemplu

EXPO.IN
```
pop iarna 1998 2000000 L 1999 1
popa ploaia 2002 5000000 E 2001 2 Verona Italia
rus copii 1999 1500000 N 2000 3 Brasov
popescu tu 2000 2500000 L 2000 1
```

LISTA1.OUT
```
Locala: pop iarna 1999 1
Locala: popescu tu 2000 1
Nationala: rus copii 2000 3 Brasov
Europeana: popa ploaia 2001 2 Verona Italia
```

LISTA2.OUT
```
rus copii 1500000
pop iarna 2000000
popescu tu 2500000
popa ploaia 5000000
```

LISTA3.OUT
```
pop iarna L
popescu tu L
```

16.5. Soluțiile problemelor rezolvate

16.5.1. Agenda

Datele referitoare la persoane se păstrează într-o structură de tip înregistrare de forma:

```
type cun=record
           nume,tel:string
         end;
```

La o rulare a programului datele se preiau din fişierul de intrare **AGENDA.IN** şi se depun în tabloul *a* de tipul: **array**[1..100] **of** cun. Toate prelucrările cerute în enunț se vor realiza asupra tabloului *a*. La finalul execuției programului se formează un fişier temporar **AGENDA.OUT** care conține tabloul a după actualizări. Fişierul de intrare **AGENDA.IN** se şterge, iar fişierul **AGENDA.OUT** se redenumeşte în **AGENDA.IN**, astfel încât la următoarea rulare a programului datele de intrare să se găsească cu conținutul actualizat.

Numărul înregistrărilor din tabloul *a* este *n*.

Citirea datelor din fişierul de intrare şi memorarea lor în tabloul *a* se face în modul următor:

- citim o linie din fişierul de intrare într-un şir de caractere *s*;
- deoarece fiecare număr de telefon începe cu caracterul '0' căutăm poziția acestuia în şirul de caractere;

- copiem caracterele de la începutul şirului de caractere (fără spaţiul despărţitor dintre nume şi numărul de telefon) în câmpul `nume` al elementului curent din şirul *a*;
- ştergem aceste caractere (inclusiv spaţiul) din şirul de caractere;
- ceea ce a rămas în şirul de caractere este numărul de telefon.

Prima acţiune posibilă de solicitat de elev este afişarea datelor din fişier (memorate deja în tabloul *a*).

Introducerea unei persoane noi în agendă se realizează astfel:
- citim numele şi numărul de telefon pe care dorim să le înregistrăm în agendă;
- apelăm subalgoritmul de căutare `Cautnume(a,n,i,nume,ok)` care va returna în variabila *ok* o valoare de adevăr;
- dacă valoarea lui *ok* este *adevărat*, înseamnă că am găsit numele persoanei, adică aceasta există deja în agendă; după revenirea din apelul subprogramului `Introd` afişăm un mesaj corespunzător şi oferim posibilitatea modificării numărului de telefon (acest lucru este posibil, deoarece în variabila *i* avem indicele elementului din şirul *a*, care conţine datele persoanei respective).

Dacă s-a introdus opţiunea corespunzătoare introducerii unei noi înregistrări, în subprogramul `Meniu(a,n,opţiune)` se va selecta secvenţa de program în care se apelează subprogramul `Introd(a,n,ok,i,nume)`:

```
...
  scrie 'Numele: '
  citeşte nume
  Introd(a,n,ok,i,nume)
  dacă ok atunci
    scrie 'Persoana ',nume,' exista deja in agenda.'
    scrie 'Daca doriti sa schimbati numarul de telefon,
                                                introduceti-l.'
    scrie 'Daca nu, introduceti 0.'
    scrie 'Numar: '
    citeşte tel
    dacă tel ≠ '0' atunci            { numerele de telefon sunt stringuri }
      a[i].tel ← tel
    sfârşit dacă
  sfârşit dacă
...
```

Subalgoritmul `Introd(a,n,ok,i,nume)` începe cu apelarea altui subprogram care va căuta numele persoanei introduse. Dacă acest nume se găseşte deja în agendă, se revine din subalgoritm cu valoarea variabilei *ok* egală cu *adevărat*. În caz contrar se

citeşte şi numărul de telefon pentru a introduce persoana în agendă. Deoarece agenda conţine numele persoanelor ordonate alfabetic, subalgoritmul `Cautnume(a,n,i, nume,ok)` este astfel realizat încât va returna şi valoarea variabilei *i* care în caz de căutare cu succes reprezintă indicele elementului din şirul *a* care conţine datele căutate, iar în caz contrar reprezintă indicele elementului unde ar trebui să se afle persoana cu numărul de telefon căutat. În concluzie, această persoană nouă trebuie să ajungă pe poziţia *i*. Pentru a putea realiza acest scop, în prealabil mărim dimensiunea şirului, apoi translatăm elementele având indicele egal sau mai mare cu *i* cu o poziţie la dreapta. Apoi, scriem datele în elementul de indice *i*.

```
Subalgoritm Introd(a,n,ok,i,nume):
  Cautnume(a,n,i,nume,ok)
  dacă nu ok atunci                    { persoana nume nu există în agendă }
    n ← n + 1                                    { mărim dimensiunea şirului }
    pentru j=n,i+1 execută:            { eliberăm poziţia i, translatând restul }
      a[j] ← a[j-1]                     { elementelor cu o poziţie la dreapta }
    sfârşit pentru
    scrie 'Numarul de telefon: '
    citeşte tel
    a[i].nume ← nume                 { datele respective se copiază pe poziţia i }
    a[i].tel ← tel
  sfârşit dacă
sfârşit subalgoritm
```

Căutarea se realizează cu subalgoritmul cunoscut (vezi capitolul 6):

```
Subalgoritm Cautnume(a,n,i,nume,ok):
              { căutam nume în agendă; dacă ok=true ⇒ l-am găsit pe poziţia i }
                { dacă ok=false ⇒ nu l-am găsit, ar fi trebuit să fie pe poziţia i }
  i ← 1
  cât timp (i ≤ n) şi (a[i].nume < nume) execută:
    i ← i + 1
  dacă a[i].nume=nume atunci
    ok ← adevărat
  altfel
    ok ← fals
sfârşit subalgoritm
```

Ştergerea unui element din agendă se efectuează, dacă opţiunea de actualizare specifică codul pentru care se execută următoarea secvenţă din subprogramul `Meniu`. Apelarea subprogramului `Şterg(a,n,ok,nume)` se finalizează fie cu ştergerea elementului corespunzător din şir, fie cu un mesaj pe baza valorii parametrului *ok*.

```
...
  scrie 'Numele: '
  citeşte nume
  Şterg(a,n,ok,nume)
  dacă nu ok atunci
    scrie 'Persoana ',nume,' nu exista in agenda.'
  sfârşit dacă
...
```

Subalgoritmul `Şterg(a,n,ok,nume)` apelează la rândul său subalgoritmul `Cautnum(a,n,i,nume,ok)` în mod similar cu subalgoritmul `Introd(a,n,ok,i, nume)`.

```
Subalgoritm Şterg(a,n,ok,nume):
  Cautnume(a,n,i,nume,ok)
  dacă ok atunci                          { persoana nume există în agendă }
    pentru j=i,n-1 execută:               { eliminăm elementul de pe poziţia i, }
      a[j] ← a[j+1]     { translatând restul elementelor cu o poziţie la stânga }
    sfârşit pentru
    n ← n - 1                                  { micşorăm dimensiunea şirului }
  sfârşit dacă
sfârşit subalgoritm
```

Căutarea numărului de telefon este identică cu cea a numelui, diferă doar câmpul analizat din elementele şirului.

16.5.2. Expoziţie

Se creează structura:

```
type str20=string[20];
     tablou=record
              nume,titlu:str20;
              an:Integer;
              pret:str20;
              case expo:(L,N,E) of
                L:(anl:Integer; prl:Char);
                N:(ann:Integer; prn:Char; loc:str20);
                E:(ani:Integer; pri:Char; loci,tara:str20)
              end;
     sir=array[1..20] of tablou;
```

Datele le vom prelua din fişierul de intrare şi vom completa fiecare înregistrare în funcţie de variantele sale. Şirul datelor (*a*) are *m* elemente.

Crearea primei liste cerute în problemă se realizează după valorile câmpului selector *expo*. Se parcurge tabloul *a* şi se aleg doar acele elemente care au câmpul selector egal cu L. Se procedează analog pentru celelalte două tipuri, identificate cu N şi E.

Crearea celui de-al doilea fişier de ieşire presupune ordonarea crescătoare a înregistrărilor din tablou după câmpul *preț*. Se poate folosi oricare algoritm de ordonare cunoscut.

Pentru păstrarea ordinii alfabetice în cazul celei de-a treia cerințe a problemei se realizează o ordonare crescătoare a înregistrărilor după câmpul *nume*, apoi se completează cel de-al treilea fişier de ieşire cu artiştii care au obținut premiul 1, specificând şi tipul expoziției.