VALIDAREA DATELOR IN PASCAL

Prin validare intelegem operatia de verificare a corectitudinii datelor. O data se considera corecta daca respecta o serie de conditii aritmetice sau logice prestabilite. Validarea datelor presupune precizarea urmatoarelor elemente:

- Conditiile de verificare
- Actiunile care trebuie executate cand data este corecta sau incorecta
- Modul de semnalare a erorilor si structura mesajelor
- Modul de corectie a erorilor
- Modul de reintroducere a articolelor dupa corectia acestora

Validarea se poate face la nivel de camp (un camp respecta propriile sale conditii), la nivel de articol (respectarea unor relatii intre campurile aceluiasi articol), la nivelul mai multor articole (respectarea unor relatii intre campurile unor articole diferite, completitudinea pe pachete de documente, corectitudinea unor totaluri pe pachete) sau la nivel de fisier (completitudine, corectitudinea unor totaluri pe fisier).

Cele mai importante validari la nivel de campuri si la nivel de articol se refera la:

- Existenta
- Natura
- Lungime
- Semn
- Apartenenta la o multime sau lista de valori
- Respectarea unor corelatii aritmetice sau logice intre campuri

Daca o data nu indeplineste toate conditiile stabilite, se considera eroare si se solicita reintroducerea ei. Introducerea se poate repeta, fie pana se obtine o data corecta, fie pana cand se depaseste un numar prestabilit de reintroduceri.

Cand un camp este corect, se trece la inroducerea si validarea urmatorului camp. Cand articolul este corect, se scrie in suportul extern si se trece la urmatorul articol.

Figura urmatoare prezinta o secventa de principiu de validare a unui camp dupa mai multe conditii:

Pentru a construi structura repetitiva, necesara reluarii introducerii campului, se utilizeaza o variabila *semafor* (booleana) **ER**, care ia valoarea **1**, daca a fost eroare (nu s-a indeplinit un criteriu de validare din multimea **C1, C2, …, Cn**), sau **0**, in caz contrar.

1.VALIDAREA EXISTENTEI

O data se considera *existenta* daca in campul corespunzator ei nu se introduce doar **<ENTER >.** Verificarea existentei se realizeaza in functie de in functie de semnificatia tastei **ENTER** la citirea diverselor tipuri de date:

- Pentru date numerice- rol de separator,fiind ignorata la citire

- Pentru date de tip caracter- furnizeaza codul ASCII al caracterului **CR** (#13)
- Pentru date de tip **STRING**- rol de terminator

Analiza presupune ca datele sunt introduse camp cu camp, cu procedura READLN. Avand in vedere ca la citirea datelor numerice, **<ENTER>** este ignorat si ca la citirea intr-o variabila de tip CHAR a lui **<ENTER>** se solicita un nou **<ENTER>** , rezulta ca, indiferent de tipul datei, singura posibilitate de verificare a existentei este citirea in variabile STRING, la care numai prin apasarea tastei **ENTER** se genereaza transferul sirului vid. Daca LENGTH(sir)=0 (sau sir[0]= #0), se considera ca data nu a fost introdusa. Daca valoarea introdusa este diferita de sirul vid se va proceda astfel:

- Pentru date *numerice*, valoarea STRING va fi convertita cu procedura **VAL**, definita in unit-ul **SYSTEM**, care se apeleaza astfel:**VAL(s,n,cod_er)**

S este variabila STRING care va fi convertita, **n** este variabila numerica in care s va depune rezultatul conversiei, iar **cod_er** e o variabila de tip INTEGER care va contine valoarea 0, daca conversia sin ASCII in binar s-a realizat fara eroare, sau pozitia in cadrul sirului a caracterului care nu a putut fi convertit (caz in care valoarea lui **n** nu se modifica).

Exemplu:

```
VAR
  Cod: WORD; cods: STRING[5]; cod_er: INTEGER; er: BOOLEAN;
BEGIN
  REPEAT
        Er:=false; Write[`Cod: `]; Readln (cods);
        IF cods [0]= #0 THEN
              BEGIN er:=true;Writeln (`>>Nu ati inrodus valoare!`) End
        ELSE Val (cods,cod,cod_er);
  UNTIL NOT er;
END
```

- Penru date de tip *caracter*, valoare STRING[1] citita va fi atribuita variabilei de tip CHAR.

Exemplu:

```
VAR
        Sex:CHAR; sexs:STRING[1]; er:BOOLEAN;
BEGIN
        REPEAT
              Er:=false; Write(`sex:`);Readln(sexs)
              IF sexs[0]=#0 THEN
                          BEGIN er:=true; Writeln(`<<Nu ati introdus
                          valoarea!`)       END
              ELSE sex:=sexs[1];
        UNTIL NOT er;
        END
```

- Pentru date de tip *sir de caractere* nu este necesara alta prelucrare.

Exemplu:

```
VAR
        Nume:STRING[30];er:BOOLEAN;
BEGIN
        REPEAT
                Er:=false; Write (`Nume:`); Readln (nume);
                IF Length (nume)=0 THEN
                                BEGIN er:=true; Writeln(`Nu ati introdus
                                valoarea!`) END
        UNTIL NOT er;
END.
```

2.VALIDAREA LUNGIMII

Lungimea sirului extern introdus de la tastatura se poate determina numai daca citire se face in variabile STRING. Dupa introducere, se testeaza lungimea sirului efectiv, determinata prin functia LENGTH sau preluata din octetul 0. Daca lungimea nu indeplineste conditia impusa (de regule,sa fie egala cu o valoare prestabilita) data se considera eronata. Daca data este corecta, se va proceda similar validarii de existenta. De fapt, acesta este un caz particular al validarii de lungime: lungimea datei sa fie diferita de 0.

- Pentru date *numerice* :

```
VAR
     Cod:WORD; cods:STRING[5]; cod_er:INTEGER; er:BOOLEAN;
CONST
        I_corecta=5;
BEGIN
        REPEAT
                Er:=false;Write(`cod:`);Readln (cods);
                IF Ord (cods[0])<>I_corecta THEN
                        BEGIN er:=true; Writeln(`Lungime eronata!`)
                        END
                ELSE Val (cods,cod,cod_er);
        UNTIL NOT er
END.
```

- Pentru date de tip *caracter*:

```
VAR
Sex:CHAR; sexs:SRING[1];er:BOOLEAN
BEGIN
REPEAT
        Er:=false; Write(`sex:`);Readln(sexs)
                IF sexs[0]<>#1 THEN
                   BEGIN er:=true; Writeln(`Lungime eronata!`)        END
        ELSE sex:=sexs[1];
UNTIL NOT er;
END
```

- Pentru date de tip *sir de caractere*:

```
VAR
Nume:STRING[30];er:BOOLEAN;
CONST
I_corecta:=15
BEGIN
REPEAT
        Er:=false; Write (`Nume:`); Readln (nume);
        IF Length (nume)<>I_corecta THEN
                        BEGIN er:=true; Writeln(`Lungime eronata!`)
                        END
        UNTIL NOT er;
END.
```

3.VALIDAREA NATURII

Datele pot avea urmatoarele naturi:

- Numerica
- Alfanumerica
- Alfabetica

In campul extern al datei numerice pot aparea semnul, cifrele 0…9, punctul zecimal si caracterele cu rol de spatiu. In campul extern al datei alfabetice pot aparea caracterele A-Z,a-z, spatiul si, eventual, alte caractere (de exemplu `-`). Pentru datele numerice si alfabetice se pot construi proceduri de validare a naturii, cu toate ca limbajul PASCAL nu poseda instructiuni dedicate unei astfel de operatii.

1.*Validarea de numericitete* se realizeaza fie direct, prin procedurile de citire, fie utilizand procedura de conversie **VAL**.

a)*Validarea directa prin citire* se bazeaza pe faptul ca procedurile Read/Readln genereaza eroare si intrerup executia programului, daca in timpul conversiei sirului, introdus de la terminal se depisteaza un caracter care nu face parte din multimea admisa pentru tipurile numerice.

```
VAR
    Cod:WORD; er:BOOLEAN;
BEGIN
        REPEAT
                Er:=false;Write(`cod:`);{$I-}Readln (cod); {$I+}
                IF IOResult<>0 THEN
                        BEGIN er:=true; Writeln(`>>Cod numeric!`)
                        END
        UNTIL NOT er;
END.
```

b)*Validarea prin conversii proprii* presupune introducerea datei numerice intr-o variabila de tip STRING, urmata de conversia cu procedura **VAL**. Dupa executie se testeaza parametrul **cod_er** al acesteia.

```
VAR
    Cod:WORD; cods:STRING[5]; cod_er:INTEGER; er:BOOLEAN;
BEGIN
    REPEAT
        Er:=false;Write(`cod:`);Readln (cods);Val(cods,cod,cod_er);
        IF cod_er<>0 THEN
            BEGIN er:=true; Writeln(`>>Cod numeric!`)
            END
    UNTIL NOT er
END.
```

2.*Validarea naturii alfabetice* se realizeaza prin verificarea naturii fiecarui caracter din sirul citit (prin expresii relationale sau prin expresii cu multimi).

```
VAR
Nume:STRING[30]; er:BOOLEAN;
CONST
Alfabet:=[`A`..`Z`, `a`..`z`,` `,`-`];
BEGIN
REPEAT
    Er:=false; Write(`Nume: `); Readln(nume);
    FOR i:=1 TO Length(nume) DO
        IF NOT (nume[i] IN alfabet) THEN er:=true;
    IF er THEN Writeln(`>>Data nealfabetica!`)
UNTIL NOT er;
END.
```

4.VALIDAREA APARTENENTEI LA O MULTIME

Verificarea apartenentei unei *date numerice* la o multime prestabilita de valori se realizeaza fie prin expresii cu multimi (utilizand operatorul **IN**), daca multimea este ordinala si se poate defini un literal de tip **SET**, fie prin expresii relationale, daca multimea este reala sau nu se poate defini o constanta **SET**.

```
VAR
    Cod_grupa:WORD; vb,er:BOOLEAN;
CONST catalog=[201..250];
BEGIN
    REPEAT
        Er:=false; Write(`cod grupa:`);Readln(cod_grupa);
        IF NOT (cod_grupa IN catalog) THEN
            BEGIN er:=true;Writeln(`>>Cod grupa eronat!`) END
    UNTIL NOT er;
END.
```

Pentru *date de tip STRING*, verificarea se poate realiza prin cautarea intr-o tabela generata prin program (de exemplu, o constanta de tip vectori de siruri).

```
VAR sectie:STRING[15]; i:BYTE; vb,er:BOOLEAN;
```

```
CONST nomenclator:ARRAY[1..4] OF STRING[15]=(`Informatica`, `Statistica`,
`Cibernetica`,`Matematica`);
BEGIN
        REPEAT
                Er:=false; vb:=false; Write(`Sectia:`);Readln(sectie)
                FOR i=1 TO Length(sectie) DO sectie[i]:=UpCase(sectie[i]);
                i:=1;
                WHILE (i<=4) AND NOT vb DO
                IF sectie=nomenclator[i] THEN vb:=true ELSE i:=i+1;
                IF NOT vb THEN
                BEGIN er:=true;Writeln(`>>Sectie inexistenta !`) END
UNTIL NOT er;
END.
```

Pentru *date de tip CHAR* pot fi folosite expresii cu multimi (operatorul **IN**).

```
VAR
        tip_bursa:CHAR;vb,er:BOOLEAN;
BEGIN
        REPEAT
                Er:=false;
                Write (`Tipul bursei (1/2/S/M):`);
                Readln(tip_bursa):
                IF NOT(tip_bursa IN [`1`,`2`,`S`,`M`]) THEN
                        BEGIN
                                Er:=true;
                                Writeln (`>>Tip bursa eronat!`)
                        END
        UNTIL NOT er;
END.
```