Bazele programării I
Structuri de date. Mulțime

Tipul de date Mulțime

Tipul mulțime este un tip de date structurat, total diferit de celelalte tipuri structurate. Mulțimea este o totalitate de obiecte distincte.

Obiectele mulțimii se numesc **elemente**.

Proprietățile mulțimii:

- **Toate elementele mulțimii aparțin unui domeniu;**

  Mulțimea tuturor numerelor întregi este domeniu pentru orice mulțime de numere naturale.

- **Orice element al mulțimii are numai două alternative: aparține sau nu aparține mulțimii respetive;**

  Mulțimile {1,2,7} și {1,1,2,7,7} sunt identice.

- **Ordinea enumerării element. mulțimii nu are importanță.**

  Mulțimile {1,2,7}, {2,7,1} și {7,2,1,1,7} sunt identice.

Reprezentarea datelor ca mulţime

Într-un algoritm mulțimile de obiecte pot fi reprezentate sub diferite forme de structuri de date:

- Tablouri;
- Fișiere;
- Structuri de date dinamice;
- Mulțimi.

Datele pot fi reprezentate ca date de tip mulțime numai în acele situații când se **respectă cele trei proprietăți ale mulțimii**.

Definirea tipului de date mulțime

```
Type
     <Nume_tip> = Set of <Nume_tip_baza>
```

Nume_tip_baza – este tip de date ordinal (Char, Boolean, enumerare, interval de Natural, interval de Integer, interval de Char).

Domeniul de definiție a tipului mulțime reprezintă **mulțimea tuturor submulțimilor** tipului de bază, inclusiv mulțimea vidă.
Dacă tipul de bază are N valori, atunci domeniul de definiție va conține **2^n valori**.

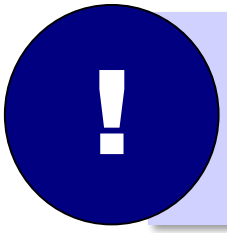
Exemplu

```
Type
   Oras = (Balti,Floresti,Soroca)
   M_Oras = Set of Oras
Var
   X: M_Oras
```

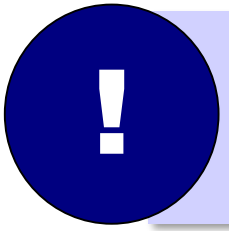
Variabila X poate primi $2^3 = 8$ valori diferite.

```
[ ]
[Balti]
[Floresti]
[Soroca]
[Balti, Floresti]
[Balti, Soroca]
[Floresti, Soroca]
[Balti, Floresti,
Soroca]
```

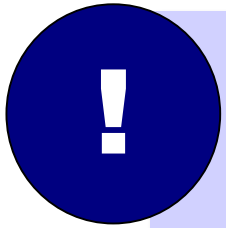
Reprezentarea în memorie a variabilelor de tip mulțime



Variabilele de tip mulțime se reprezintă în memoria operativă pe **N biți**.



O variabilă de tip mulțime se alocă pe **(N div 8)+1** octeți.



Fiecare element posibil al mulțimii se reprezintă pe **un bit**. Dacă elementul aparține mulțimii, atunci bitul respeciv este poziționat pe 1, în caz contrar – pe 0.

Exemplu

```
Type
   Oras = (Balti,Floresti,Soroca)
   M_Oras = Set of Oras
Var
   M1: M_Oras
   M2: M_Oras
   M3: M_Oras
   M4: M_Oras
```

```
M1:=[Balti]
M2:=[Balti,Soroca]
M3:=[Balti,Floresti,
     Soroca]
M4:=[]
```

1	0	0
1	0	1
1	1	1
0	0	0

Exemplu

```
Type
   Cifre = 0..9
   M_Cifre = Set Of Cifre
Var
   Impare: M_Cifre
   Pare: M_Cifre
```

Variabilele Impare, Pare poate primi $2^{10} = 1024$ valori diferite.

```
Impare:= [1,3,5,7,9]
Pare:= [0,2,4,6,8]
```

0	1	0	1	0	1	0	1	0	1

1	0	1	0	1	0	1	0	1	0

Exemplu

```
Type
   Semne = Set of Char

Var
   X: Semne
```

Variabila X poate primi 2^{256} valori diferite.

```
X:= ['a','e','i','o','u']
X:= ['b','c','d','f','g']
```

Operații cu date de tip mulțime

```
Var
   Par: M_Cifre
   Impar: M_Cifre
   Litere: Semne
   Cifre: M_Cifre
   Este: Boolean
```

Atribuire

Variabilele de tip mulțime pot fi inițializate doar prin atribuire.

```
Par:= [0,2,4,6,8]
Impar:= [1,3,5,7,9]
Litere:= ['a'..'z']
```

Operații cu date de tip mulțime

Reuniunea a două mulțimi "+"

Este mulțimea formată din elementele comune și necomune ale celor două mulțimi.

```
Cifre:= [2,5,8,9] + [4,2,9]
```

Cifre = [2,4,5,8,9]

Intersecția a două mulțimi "*"

Este mulțimea formată din elementele comune a două mulțimi.

```
Cifre:= [2,5,8,9] * [4,2,9]
```

Cifre = [2, 9]

```
Cifre:= [2,5,7] * [4,6,9]
```

Cifre = []

Operații cu date de tip mulțime

Diferența a două mulțimi "–"

Este mulțimea formată din elementele primei mulțimi, care nu aparțin celei de-a doua mulțimi.

```
Impar:= [1..9] - [2,4,6,8]
```

Impar = [1,3,5,7,9]

Operații relaționale

Datele de tip mulțime pot să apară în expresii relaționale.

```
Este:= [1,2,3] = [3,2,1]
```

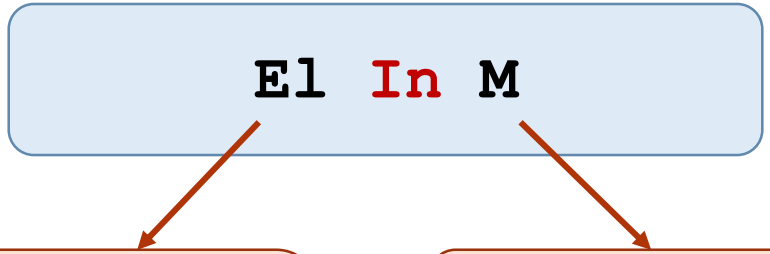
Este = True

```
Este:= [1..9] >= [2,4,6]
```

Este = True

Selectarea unei componente prin testul de apartenență

Elementele unei date de tip mulțime nu pot fi accesate direct, dar se permite verificarea apartenenței unui element mulțimii respective, folosind operatorul relațional **In**.



El – este o expresie de același tip ordinal cu tipul de bază al mulțimii M

M – este o mulțime

```
C:= 'D'
Este:= C In ['A'..'F']
```

Este = True

Introducerea mulțimilor

```
Type
   Tb = <tip ordinal care permite operaţia
          de citire>
   Tm = Set of <Tb>
Var
   El: Tb
   M: Tm
```

Datele de tip mulţime nu pot fi nici citite, nici scrise.

```
Begin
    M:= []
    For, While, Repeat
      Read…(El)
      M:= M + [El]
    End
End
```

Afișarea mulțimilor

```
Type
   Tb = <tip ordinal care permite operaţia
          de afisare>
   Tm = Set of <Tb>
Var
   El: Tb
   M: Tm
```

Se organizează un ciclu care verifică dacă fiecare element posibil aparține cu siguranță mulțimii

```
Begin
   For El:=<primul element Tb> To
     <ultimul element Tb>  Step 1
    If El In M Then
      Write…(El)
    End
   End
End
```

Exemplu

De la tastatură se citește un număr natural. Să se afișeze cifrele din care este format acest număr (se vor afișa în ordine crescătoare cifrele distincte ale numărului).
De exemplu, pentru numărul 21514, rezultatul va fi – 1,2,4,5.

1. Introducerea numărului natural;

2. Determinarea mulțimii cifrelor din care este format numărul;

3. Afișarea cifrelor din care este format numărul.

Exemplu

```
Type
   Tb = 0..9
   Tm = Set of Tb
Var
   Num: Natural
   M_Cifre: Tm
   Cifra: Tb
Begin
   ReadNat(Num)
   M_Cifre:= []
   While Num <> 0 do
      Cifra:= Num mod 10
      M_Cifre:= M_Cifre + [Cifra]
      Num := Num div 10
   End
```

introducerea numărului natural

se va forma mulţimea cifrelor 0..9 în variabila M_Cifre

Exemplu

```
   For Cifra:= 0 to 9 step 1
      If Cifra In M_Cifre Then
         WriteNat(Cifra)
      End
   End
End
```

Afișarea cifrelor din care este format numărul

Exemplu

Să se elaboreze un algoritm care citește două numere naturale și afișează cifrele care se întâlnesc în cel puțin unul din ele.

1. Introducerea numerelor naturale;

2. Determinarea mulțimii cifrelor distincte din ambele numere;

- **2.1.** Determinarea mulțimii cifrelor prezente în Num1;
- **2.2.** Determinarea mulțimii cifrelor prezente în Num2;
- **2.3.** Determinarea cifrelor distincte din ambele numere;

3. Afișarea cifrelor care se întâlnesc în cel puțin unul din numere.

Exemplu

```
Type
   Tb = 0..9
   Tm = Set of Tb
Var
   Num1, Num2: Natural
   M_Cifre1, M_Cifre2, M_cifre3: Tm
   Cifra: Tb
Procedure Formare(Num: Natural, Var M_Cifre: Tm)
Var   Cifra: 0..9
Begin
      M_Cifre:= []
      While Num <> 0 do
         Cifra:= Num mod 10
         M_Cifre:= M_cifre + [Cifra]
         Num:= Num div 10
      End
End
```

Formarea mulţimii cifrelor prezente într-un număr natural

Exemplu

```
Begin
   ReadNat(Num1)
   ReadNat(Num2)
   Formare(Num1, M_Cifre1)
   Formare(Num2, M_Cifre2)
   M_Cifre3:= M_Cifre1 + M_Cifre2
   For Cifra:= 0 to 9 step 1
      If Cifra In M_Cifre3 Then
            WriteNat(Cifra)
      End
   End
End
```

Formarea mulțimii cifrelor distincte din reuniunea mulțimilor

Afișarea cifrelor care se întâlnesc în cel puțin unul din numere