

# Bazele programării I

## Tablouri unidimensionale

# Tablouri unidimensionale. Necesitate

---

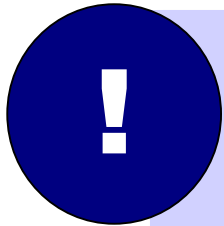
În multe probleme apare necesitatea de a lucra cu grupuri de variabile de același tip.

Pentru a păstra **notele studentului** de la sesiunea de iarnă este nevoie de un grup de 6 numere naturale (se consideră că studentul are 6 examene în sesiune).

Pentru a păstra **temperaturile zilnice ale lunii noiembrie** este nevoie de un grup de 30 numere întregi (luna noiembrie are 30 zile).

# Tablouri unidimensionale. Noțiune

---



Un grup de elemente de același tip poate fi reprezentat sub forma de **tablou unidimensional** sau **vector**.



Tabloul unidimensional constituie o colecție de date de **același tip**, care pot fi accesate prin intermediul aceluiași nume de variabilă, folosind **indici**.

# Declararea tipului de date

## Tablou unidimensional (Vector)

### **Type**

*<Nume\_tip> = **Array** <tip\_indice> **Of** <tip\_element>*



**Tip indice** – indică modalitatea de numerotare a elementelor tabloului și poate fi de orice tip ordinal.

**Tip element** – indică tipul de date a elementelor și poate fi de orice tip de date.

# Tablouri unidimensionale. Exemplu

---

*Type*

*Vector1 = Array[1..7] Of Integer*

*Var*

*x1: Vector1*



( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 ) ( 6 ) ( 7 )

# Accesarea unui element al vectorului

---



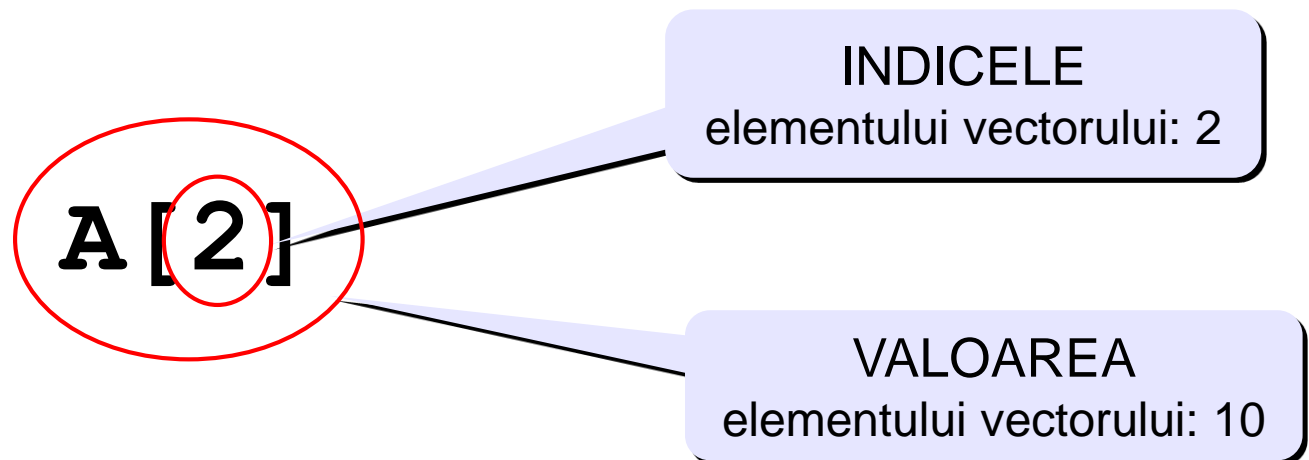
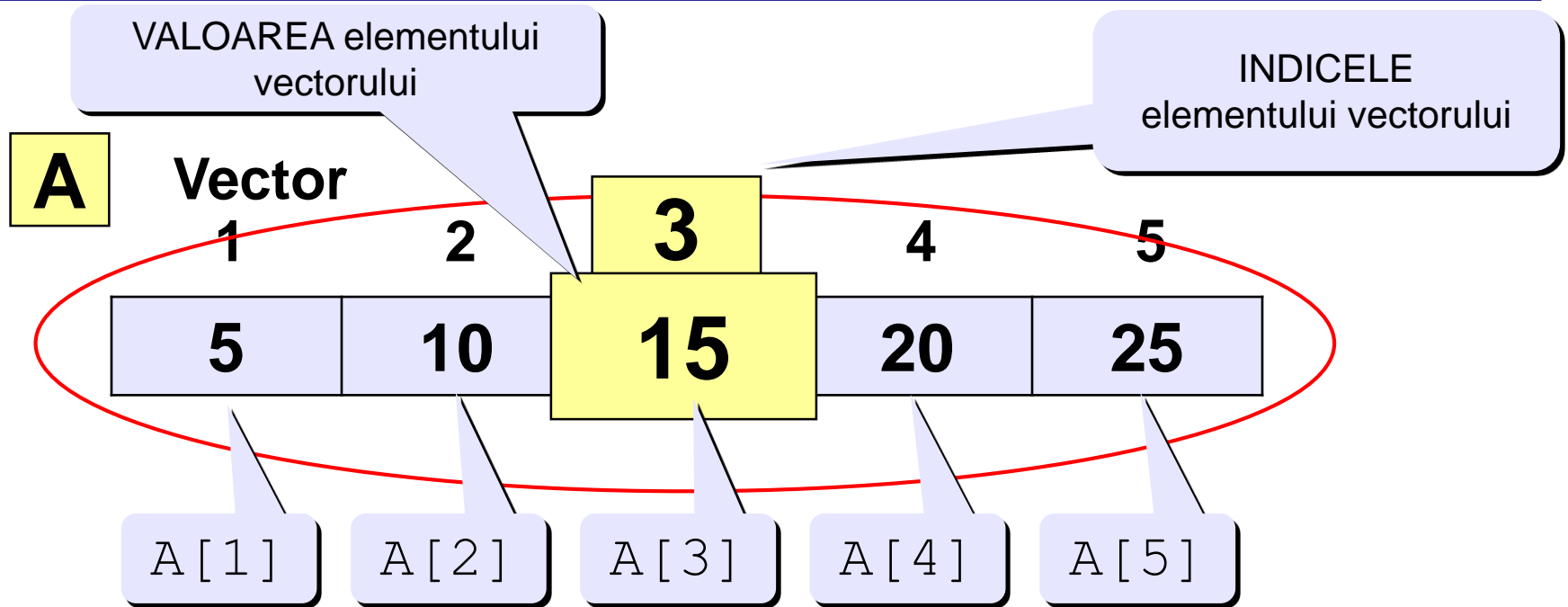
Într-un tablou unidimensional fiecărui element îi este asociat un **indice**, prin intermediul căruia elementul poate fi accesat.



Accesarea elementului se face folosind **numele variabilei** de tip *Tablou unidimensional*, urmat de o pereche de paranteze drepte între care se indică **valoarea indicelui**.

***<Nume\_variabila \_tablou>[valoare indice]***

# Tablouri unidimensionale. Exemplu

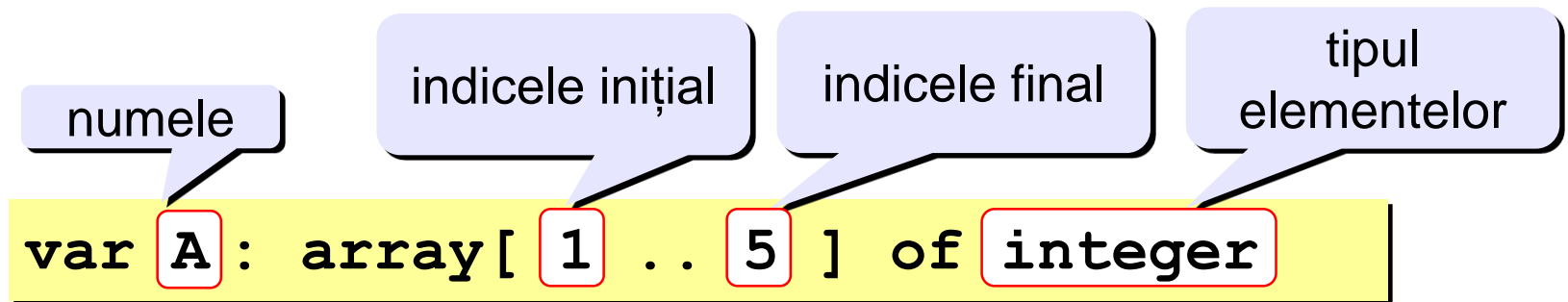


# Declararea tipului de date Vector

## Declararea vectorului

- definirea **numelui** vectorului
- definirea **tipului** vectorului
- definirea **numărului de elemente**
- alocarea **spațiului de memorie**

## Vectorul de numere întregi:



## Definirea mărimii vectorului utilizând o constantă:

```
const N=5  
var A: array[1..N] of integer
```



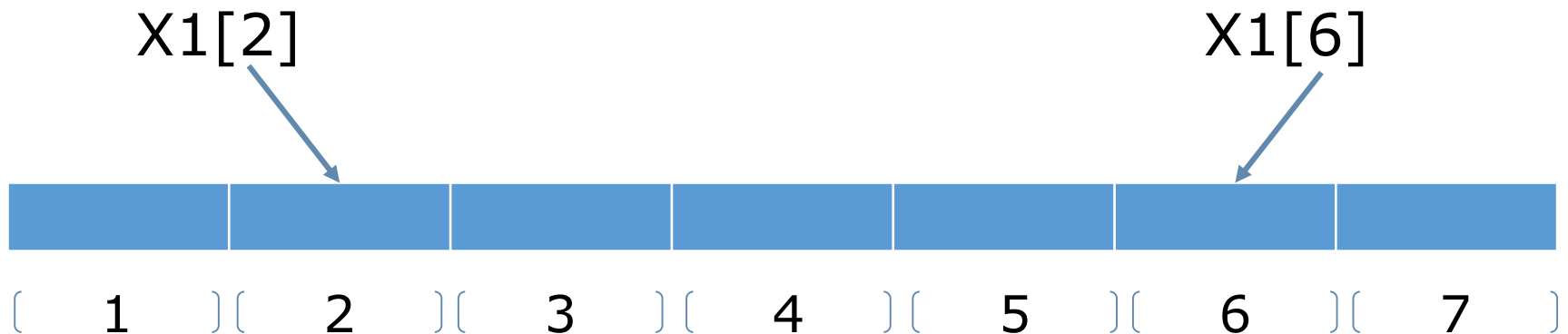
# Tablouri unidimensionale. Exemplu

*Type*

*Vector1 = Array[1..7] Of Integer*

*Var*

*x1: Vector1*



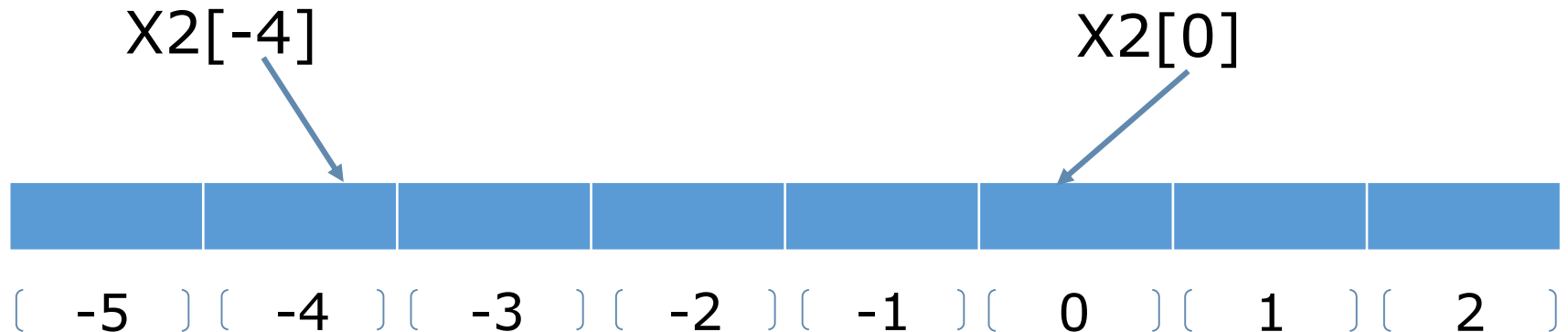
# Tablouri unidimensionale. Exemplu

*Type*

*Vector2 = Array[-5..2] Of Integer*

*Var*

*x2: Vector2*



# Tablouri unidimensionale. Exemplu

*Type*

*Lucratori = (Ion, Petru, Maria)*

*Vector3 = Array[Lucratori] Of Real*

*Var*

*Salar: Vector3*

Salar[Ion]

Salar[Maria]



# Tablouri unidimensionale. Exemplu

---

*Type*

*Vector4 = Array[Boolean] Of Natural*

*Var*

*x4: Vector4*

X4[False]

X4[True]



[

False

]

[

True

]

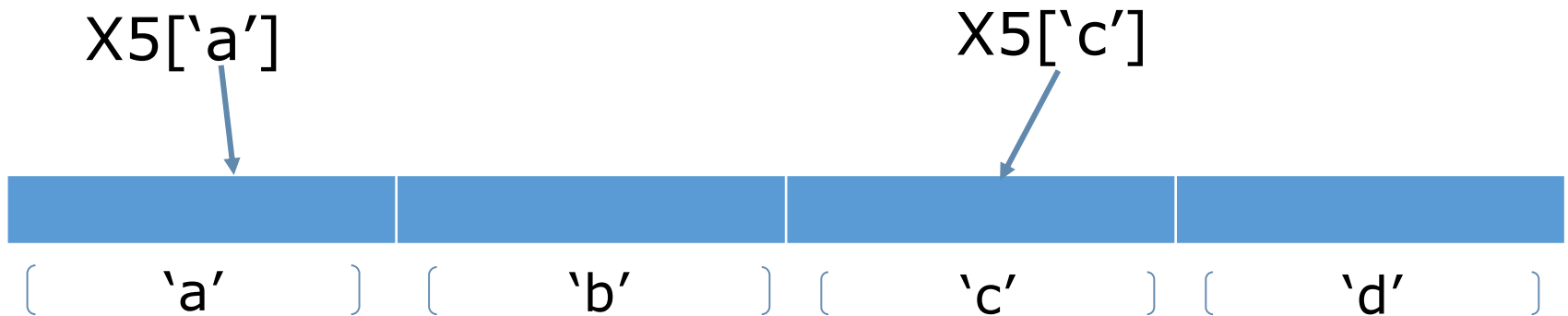
# Tablouri unidimensionale. Exemplu

*Type*

*Vector5 = Array['a'.. 'd'] Of Real*

*Var*

*x5: Vector5*



# Accesarea directă a unui element

---



O variabilă de tip vector nu poate fi **nici citită** și **nici scrisă** în întregime.



De obicei, se lucrează cu elementele vectorului. Elementele unui vector sunt **variabile obișnuite**. La ele se pot aplica toate operațiile posibile cu date de acest tip.

# Exemplu

*Type*

*Vector1 = Array[1..7] Of Integer*

*Var*

*x1: Vector1*



[ 1 ] [ 2 ] [ 3 ] [ 4 ] [ 5 ] [ 6 ] [ 7 ]

**x1[2] := 0**

**x1[5] := x1[5] + 10**

# Ce este greșit?

---

```
var A: array [1..10] of integer
```

```
...
```

```
A[5] := 4.5
```

```
var A: array ['a'..'z'] of integer
```

```
...
```

```
A['b'] := 15
```

```
var A: array [0..9] of integer
```

```
...
```

```
A[10] := 'X'
```



# Prelucrarea secvențială a elementelor tabloului

---

Deoarece o structură de tip *tablou unidimensional* conține un număr concret de elemente, pentru a le prelucra pe toate se va organiza **un proces repetitiv**, care se va repeta pentru fiecare element a acestuia.

Prelucrarea secvențială a elementelor unui vector poate fi organizată în două direcții:

- **Înainte;**
- **Înapoi.**

# Modelul de prelucrare secvențială înainte

---

Se începe cu prelucrarea **primei componente**. Succesiv vor fi prelucrate toate componentele vectorului. Ultima componentă a vectorului va fi prelucrată la sfârșit.

```
For I:= <valoarea inițială indice> To  
        <valoarea finală indice> Step 1  
  
        <Prelucrarea elementului I>  
  
End
```

# Modelul de prelucrare secvențială înapoi

---

Pentru a prelucra elementele tabloului în ordine inversă se va organiza parcurgerea elementelor tabloului, începând cu **ultimul element** până la primul.

```
For I:= <valoarea finală indice> To  
        <valoarea inițială indice> Step -1  
  
        <Prelucrarea elementului I>  
  
End
```

# Metode de lucru cu componentele vectorilor

---

*Type*

*Vector* = Array[**N1** .. **N2**] of *Integer*

*Var*

**A**: *Vector*

**N1** și **N2** reprezintă constante.

Vectroul **A** va conține (**N2-N1+1**) elemente.

# Citirea vectorului de la tastatură

*Se va aplica modelul de prelucrare secvențială a vectorului.*

```
For I:= <valoarea inițială indice>  
  To <valoarea finală indice>  
  Step 1  
    <prelucrarea elementului I>  
End
```

N1

N2

ReadInt (A[I])

```
For I:= N1 To N2 Step 1  
  ReadInt (A[I])  
End
```

# Generarea vectorului

---

Există diferite metode de generare a componentelor tabloului.

**Exemplu 1.** *Să se formeze un vector, elementele căruia sunt primele  $N$  numere naturale.*

```
X:=1
For I:=1 to n step 1
  A[I] := X
  X:=Succ(X)
End
```

```
For I:=1 to n step 1
  A[I] := I
End
```

# Generarea vectorului

---

**Exemplu 2.** *Să se formeze un vector elementele căruia sunt primele N numere Fibonacci.*

*Șirul Fibonacci.*

***1, 1, 2, 3, 5, 8, 13, 21, ...***

```
A[1] := 1
```

```
A[2] := 1
```

```
For I:=3 to N step 1
```

```
    A[I] := A[I-2] + A[I-1]
```

```
End
```

# Afișarea unui vector

---

Afișarea elementelor unui vector poate fi realizată în 3 moduri:

- Afișarea **unui singur element** specificat;
- Afișarea **tuturor elementelor** vectorului;
- Afișarea elementelor care **posedă o careva proprietate**.



# Afișarea elementului specificat

---

În acest caz, se accesează direct elementul specificat, folosind indicele asociat **I**.

**Write...(A[I])**

**Exemplul 1.** *Să se afișeze ultimul element al vectorului A.*

**WriteInt(A[N2])**

# Afişarea tuturor elementelor

---

Componentele vectorului pot fi afișate în ordine directă sau inversă:

- Toate elementele pe un rând;
- Fiecare element pe un rând.

# Afișarea vectorului - toate elementele pe un rând

*Se va aplica modelul de prelucrare secvențială a vectorului.*

```
For I:= <valoarea inițială indice>  
  To <valoarea finală indice>  
  Step 1  
    <prelucrarea elementului I>  
End
```

N1

N2

WriteInt(A[I])

```
For I:= N1 To N2 Step 1  
  WriteInt(A[i])  
End
```

# Afișarea vectorului - fiecare element pe un rând

*Se va aplica modelul de prelucrare secvențială a vectorului.*

```
For I:= <valoarea inițială indice>  
  To <valoarea finală indice>  
  Step 1  
    <prelucrarea elementului I>  
  
End
```

N1

N2

WriteInt(A[I])  
Writeln

```
For I:= N1 To N2 Step 1  
  WriteInt(A[I])  
  Writeln  
  
End
```

## Afișarea elementelor tabloului care posedă o careva proprietate

- Elementul este negativ ( $A[I] < 0$ );
- Elementul este pozitiv ( $A[I] > 0$ );
- Elementul este par ( $A[I] \bmod 2 = 0$ );
- Elementul este impar ( $A[I] \bmod 2 \neq 0$ );
- Elementul este egal cu valoare X ( $A[I] = X$ );
- Elementul este divizibil la 3 ( $A[I] \bmod 3 = 0$ );
- Elementul este număr prim;
- Elementul este număr perfect ș.a.

# Modelul afișării

*Se va aplica modelul de prelucrare secvențială a vectorului.  
La fiecare iterație se va verifica dacă elementul posedă proprietatea.*

```
For I:= <valoarea inițială> To  
    <valoarea finală> Step 1  
    <prelucrarea elementului I>  
End
```

N1

N2

```
If A[I]=<proprietate> Then  
    WriteInt(A[I])  
End
```

```
For I:= N1 To N2 Step 1  
    If A[I] = <proprietate> Then  
        WriteInt(A[I])  
    End  
End
```

# Afișarea elementelor pare

---

```
For I:= N1 To N2 Step 1
  If A[I] = <proprietary> Then
    WriteInt(A[I])
  End
End
```

$A[I] \bmod 2 = 0$

```
For I:= N1 To N2 Step 1
  If A[I] mod 2 = 0 Then
    WriteInt(A[I])
  End
End
```

# Modificarea elementelor vectorului

---

Pot fi 3 tipuri de modificare a conținutului tabloului:

- Modificarea **unui element**;
- Modificarea **tuturor elementelor**;
- Modificarea elementelor care posedă o careva proprietate.



# Modificarea elementului specificat

---

În acest caz, se accesează direct elementul care necesită modificare, folosind indicele asociat **I**.

$$A[I] := \dots$$

**Exemplul 1.** *Să se mărească cu 5 ultimul element al vectorului A.*

$$A[N2] := A[N2] + 5$$

# Modificarea tuturor elementelor

*Se va aplica modelul de prelucrare secvențială a vectorului înapoi.*

```
For I:= <valoarea finală indice> To  
    <valoarea inițială indice>  
    Step -1  
    <prelucrarea elementului I>  
End
```

N2

N1

A[I] := ...

```
For I:= N2 To N1 Step -1  
    A[I] := ...  
End
```

# Modificarea elementelor care posedă o proprietate

*Se va aplica modelul de prelucrare secvențială a vectorului. La fiecare iterație se va verifica dacă elementul posedă proprietatea.*

```
For I:= <valoarea inițială>  
  To <valoarea finală>  
  Step 1  
    <prelucrarea elementului I>  
End
```

N1

N2

```
If A[I]=<proprietate> Then  
  A[I] := ...  
End
```

```
For I:= N1 To N2 Step 1  
  If A[I] = <proprietate> Then  
    A[I] := ...  
  End  
End
```

# Modificarea elementelor negative

---

```
For I:= N1 To N2 Step 1
  If A[I] = <proprietate> Then
    A[I]:= <valoare modificată>
  End
End
```

$A[I] < 0$

$A[I] \text{ div } 3$

```
For I:= N1 To N2 Step 1
  If  $A[I] < 0$  Then
     $A[I] := A[I] \text{ div } 3$ 
  End
End
```

# Numărarea elementelor vectorului cu o proprietate dată

*Se va aplica modelul de prelucrare secvențială a vectorului.  
La fiecare iterație când se găsește un element care posedă  
proprietatea dată contorul se incrementează cu 1.*

Rez := 0

For I := <valoarea inițială>  
To <valoarea finală>  
Step 1

<prelucrarea elementului I>

End

N1

N2

If A[I] = <proprietate> Then

Rez := Rez + 1

End

Rez := 0

For I := N1 To N2 Step 1

If A[I] = <proprietate> Then

Rez := Rez + 1

End

End

# Numărarea elementelor

*Să se determine câte elemente divizibile prin 5 sau 7 conține vectorul.*

```
Const N = 5
```

```
Var A: array [1..N] of integer
```

```
    i, count: natural
```

```
Begin
```

```
    { completăm vectorul }
```

```
    count := 0
```

Parcurgem toate  
elementele vectorului

```
    For I:=1 to N step 1
```

```
        If (A[I] mod 5=0) or (A[I] mod 7=0) then
```

```
            count := count + 1
```

```
        end
```

```
    end
```

```
    WriteString('Divizibile prin 5 sau 7: ')
```

```
    WriteNat(count)
```

```
End
```

# Însumarea tuturor componentelor unui vector

*Se va aplica modelul de prelucrare secvențială a vectorului.  
La fiecare iterație, suma crește cu valoarea elementului  
curent.*

`S:=0`

`For I:= <valoarea inițială> To  
          <valoarea finală> Step 1`

`<prelucrarea elementului I>`

`End`

N1

N2

`S:=S+A[I]`

`S:=0`

`For I:= N1 To N2 Step 1`

`S:=S+A[I]`

`End`

# Însumarea elementelor vectorului cu o proprietate dată

*Se va aplica modelul de prelucrare secvențială a vectorului.  
La fiecare iterație, suma crește cu valoarea elementului care posedă proprietatea.*

S:=0

```
For I:= <valoarea inițială>  
    To <valoarea finală>  
    Step 1
```

<prelucrarea elementului I>

End

N1

N2

If A[I]=<proprietate> Then

S := S + A[I]

End

S:=0

For I:= N1 To N2 Step 1

If A[I] = <proprietate> Then

S := S + A[I]

End

End



# Însumarea elementelor

*Să se calculeze suma elementelor divizibile prin 3.*

```
Const N = 5
Var A: array [1..N] of integer
    i, S: integer
Begin
    { completăm vectorul }
    S := 0
    For I:=1 to N step 1
        If A[I] mod 5 = 3 then
            S := S + A[I]
        end
    end
    WriteString('Suma elementelor: ')
    WriteInt(S)
End
```

Parcurgem toate  
elementele vectorului

# Maximul și minimul unui vector

```
{ considerăm că primul element este maxim }  
for I:=N1+1 to N2 step 1  
    if A[I] > { maximul } then  
        {memorizăm noul element maxim A[I]}  
    end  
end
```

```
{considerăm că primul element este minim}  
for I:=N1+1 to N2 step 1  
    if A[I] < { minimul } then  
        {memorizăm noul element minim A[I]}  
    end  
end
```



De ce ciclul începe de la  $I=N1+1$ ?

# Determinarea existenței în vector a elementelor cu o proprietate dată (Metoda 1)

Se consideră următoarele declarații:

Type

Vector = Array[N1..N2]  
Of Integer

Var

A: Vector

I: Natural

Exist: Boolean

2	4	6	0	15	1	18
---	---	---	---	----	---	----

(1) (2) (3) (4) (5) (6) (7)



N1



N2

I ≤ N2	1 ≤ 7	2 ≤ 7	3 ≤ 7	4 ≤ 7	5 ≤ 7
Exist	F	F	F	T	T

## Căutarea liniară

Exist := False

I := N1

While (I ≤ N2) and Not Exist do

If A[I] = 0 Then

Exist := True

End

I := I + 1

End

If Exist Then

WriteString('Exista')

Else

WriteString('Nu exista')

End

## Determinarea existenței în vector a elementelor cu o proprietate dată (Metoda 2)

*Se consideră următoarele declarații:*



## Type

```
Vector = Array[N1..N2]
         Of Integer
```

## Var

**A:** Vector

# I: Natural

2	4	6	0	15	1	18
( 1 )	( 2 )	( 3 )	( 4 )	( 5 )	( 6 )	( 7 )
						
<b>N1</b>						<b>N2</b>

$I \leq N-2$	$1 \leq 7$	$2 \leq 7$	$3 \leq 7$	$4 \leq 7$
$A[I] < 0$	T	T	T	F

$$I := 1$$

**While** (I <= 7) and

```
(A[I] <> 0) do
```

**$I := I + 1$**

# End

If I ≤ 7 Then

```
WriteString( 'Exista' )
```

## Else

```
WriteString('Nu exista')
```

# End

# Determinarea existenței în vector a elementelor cu o proprietate dată (Metoda 3)

Se consideră următoarele declarații:

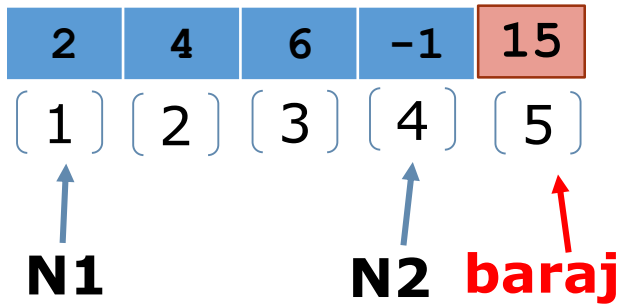
Type

Vector = Array[N1..N2+1]  
Of Integer

Var

A: Vector

I: Natural



A[I]<>15	T	T	T	T	F
I	1	2	3	4	5

## Căutarea cu baraj (înainte)

A[5] := 15

I := 1

While (A[I] <> 15) do

I:=I+1

End

If I <= 4 Then

WriteString('Exista')

Else

WriteString('Nu exista')

End

# Determinarea existenței în vector a elementelor cu o proprietate dată (Metoda 3)

Se consideră următoarele declarații:

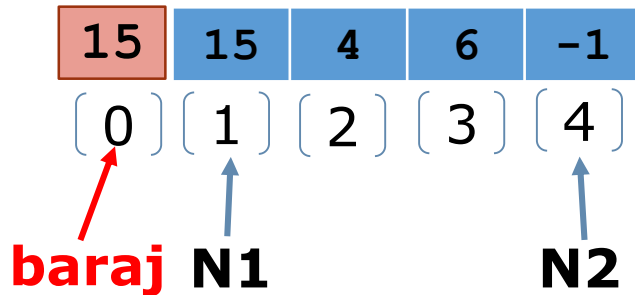
Type

Vector = Array[N1-1..N2]  
Of Integer

Var

A: Vector

I: Natural



A[I] <> 15	T	T	T	F
I	4	3	2	1

## Căutarea cu baraj (înapoi)

A[0] := 15

I := 4

While (A[I] <> 15) do

I := I - 1

End

If I >= 1 Then

WriteString('Exista')

Else

WriteString('Nu exista')

End

# Căutarea binară

**X = 7**



Vectorul în care se caută valoarea X trebuie să fie ordonat crescător.

1. Alegem elementul din mijloc  $A[\text{mijloc}]$  și îl comparăm cu X.
2. Dacă  $X = A[\text{mijloc}]$ , atunci s-a găsit elementul căutat și ieșire.
3. Dacă  $X < A[\text{mijloc}]$ , atunci căutăm în prima jumătate.
4. Dacă  $X > A[\text{mijloc}]$ , continuăm căutarea în a doua jumătate.

$x < 8$

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

$x > 4$

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

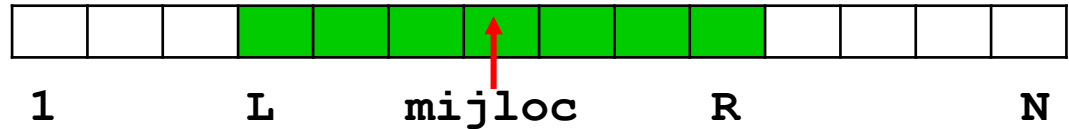
$x > 6$

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

# Căutarea binară

Var

A: **Vector**; X: **Natural**  
L, R, Mijloc: **N1 ..N2**  
Gasit: **Boolean**



```
Gasit:= false
```

```
L:= N1
```

```
R:= N2 {limite: căutăm de la A[N1] până la A[N2]}
```

```
while (L<=R) and not Gasit do
```

```
    mijloc := (R + L) div 2
```

```
    if X = A[mijloc] then
```

```
        Gasit:=true
```

```
    end
```

```
    if X < A[mijloc] then R:=mijloc - 1 end
```

```
    if X > A[mijloc] then L:=mijloc + 1 end
```

```
end
```

```
if Gasit then WriteString('Exista')
```

```
else          WriteString('Nu exista')
```

```
end
```

indicele elementului  
din mijloc

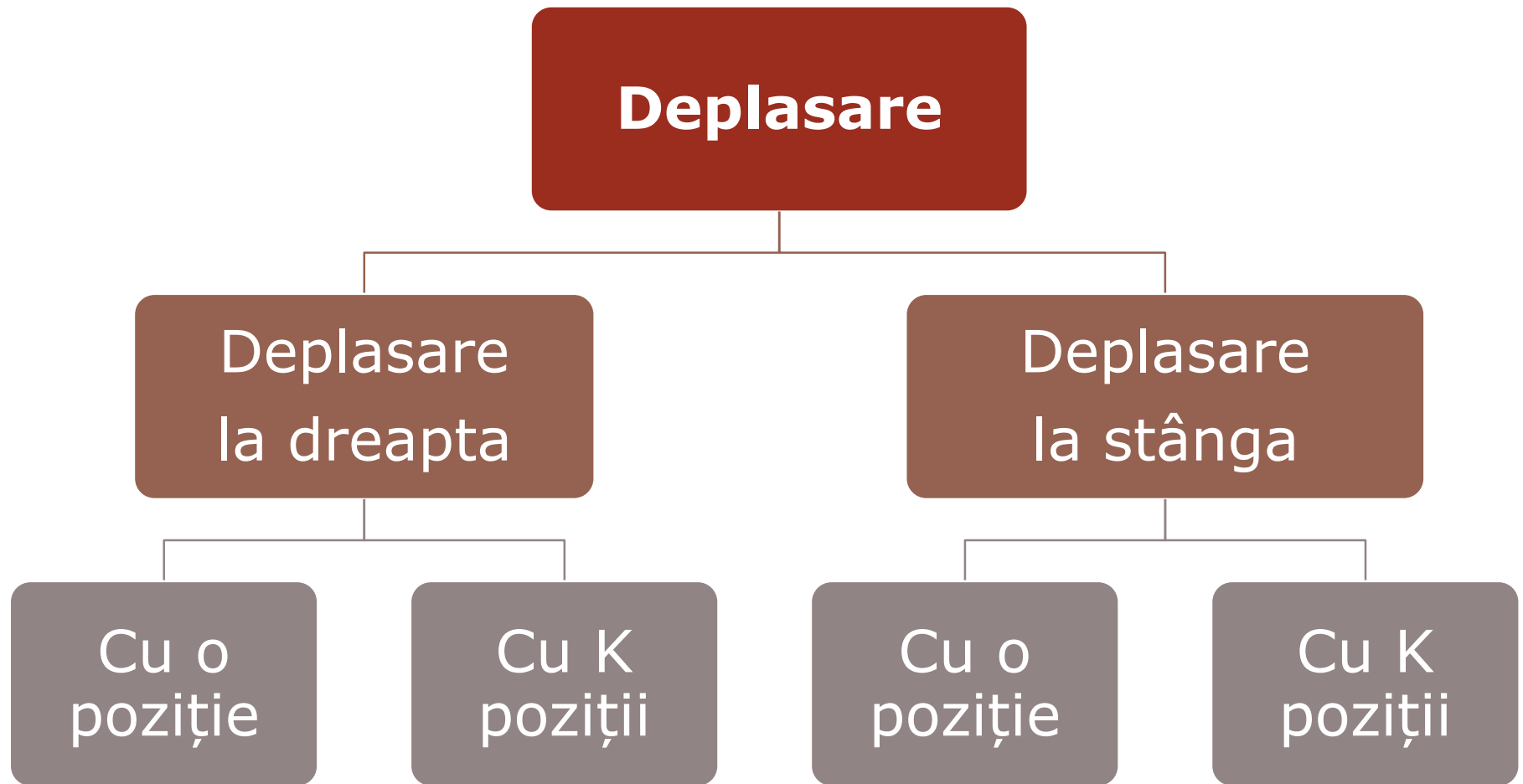
am găsit

Schimbăm  
limitele



# Algoritmi de deplasare a elementelor vectorilor

---



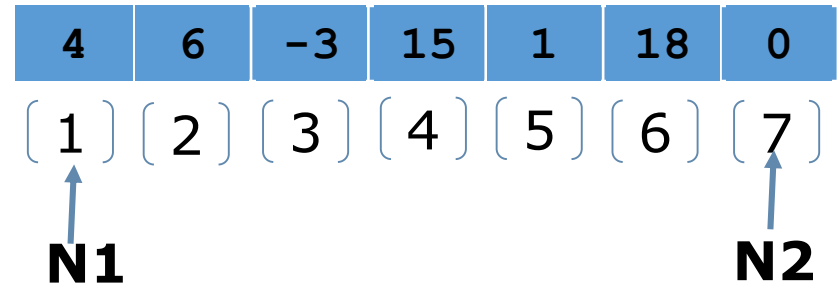
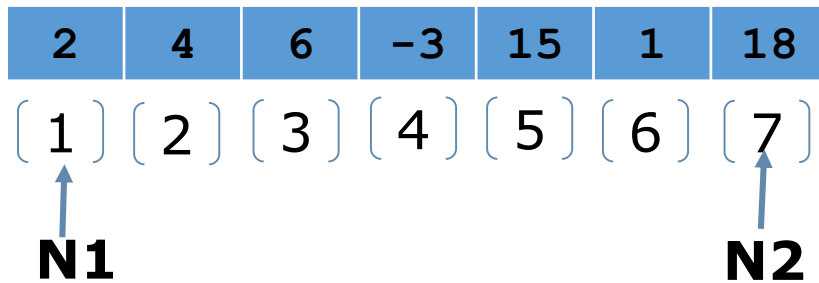
# Deplasare la stânga

Type

Vector = Array[N1..N2] Of Integer

Var

A: Vector; I: Natural



**Deplasare la stânga cu o poziție**

```
For I:=N1 to N2-1 step 1
  A[I] := A[I+1]
End
A[N2] := 0
```

**Deplasare la stânga cu k poziții**

```
For j:=1 to k step 1
  For I:=N1 to N2-1 step 1
    A[I] := A[I+1]
  End
  A[N2] := 0
End
```

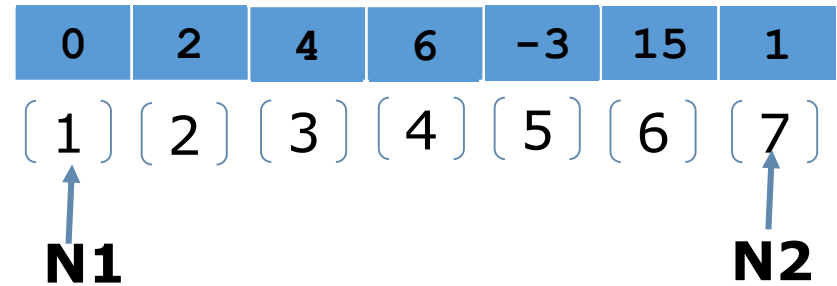
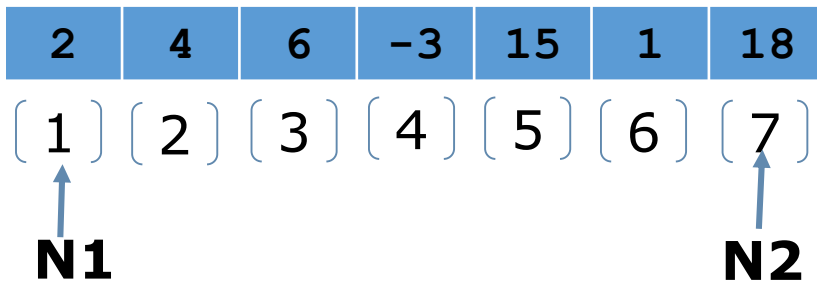
# Deplasare la dreapta

Type

Vector = Array[N1..N2] Of Integer

Var

A: Vector; I: Natural

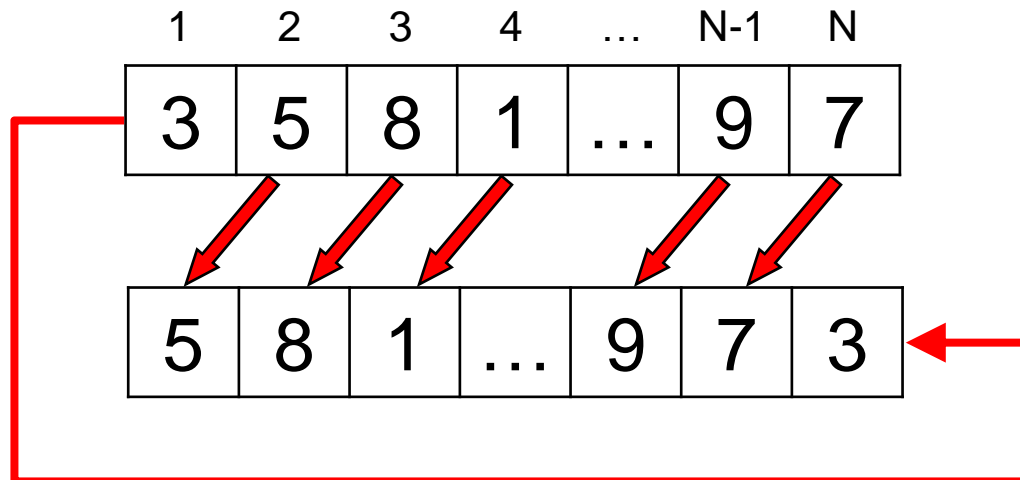
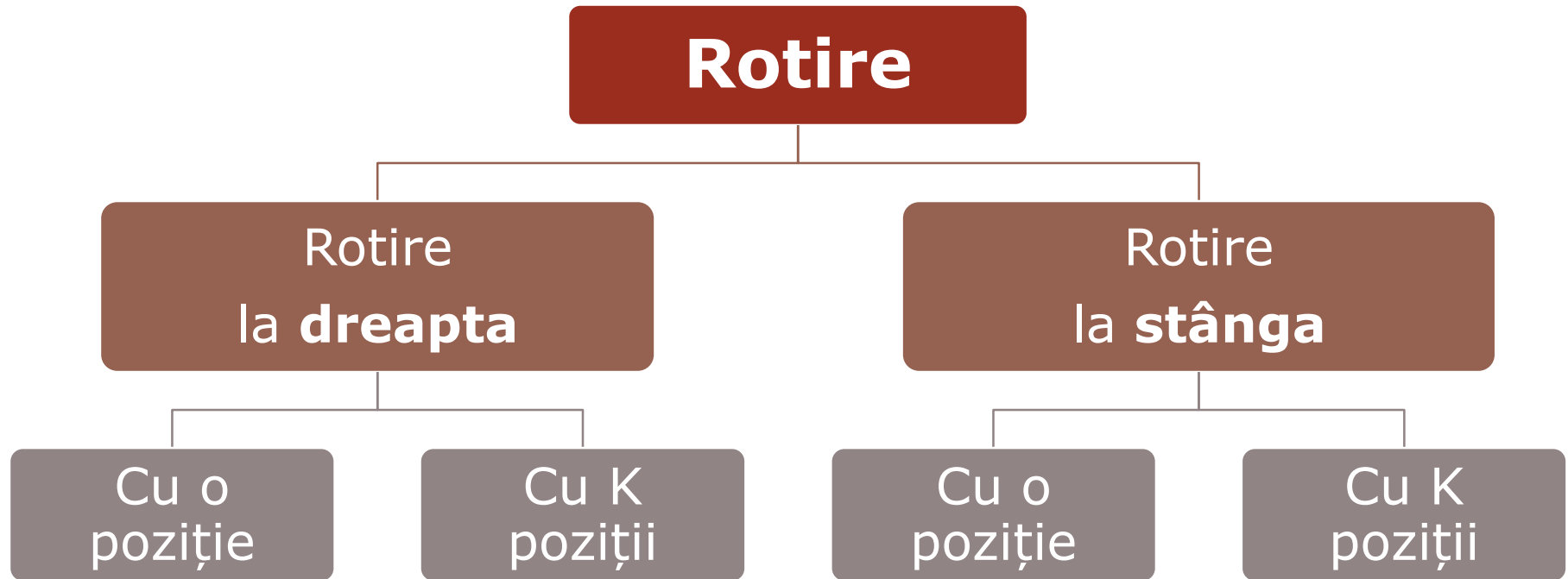


Deplasare la dreapta cu o poziție Deplasare la dreapta cu k poziții

```
For I:= N2 to N1+1 step -1
  A[I] := A[I-1]
End
A[N1] := 0
```

```
For j:=1 to k step 1
  For I:=N2 to N1+1 step -1
    A[I] := A[I-1]
  End
  A[N1] := 0
End
```

# Algoritmi de rotire a elementelor unui vector



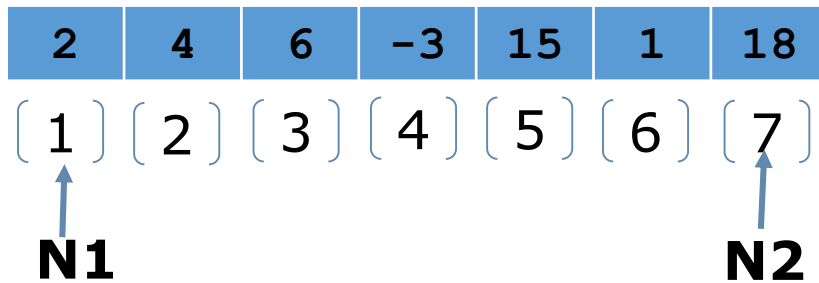
# Rotire la stânga

Type

Vector = Array[N1..N2] Of Integer

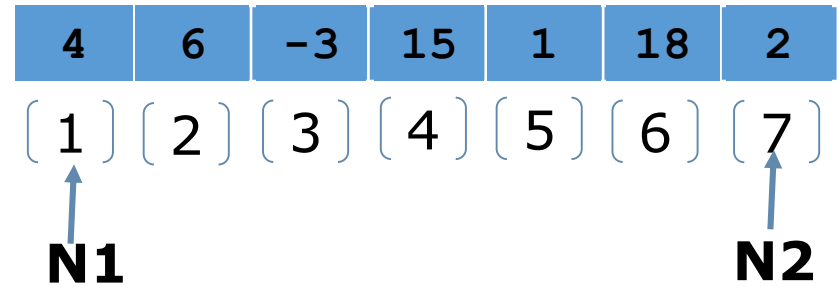
Var

A: Vector; I: Natural



**Rotire la stînga cu o poziție**

```
X:=A[N1]
For I:= N1 to N2-1 step 1
    A[I] := A[I+1]
End
A[N2] :=X
```



**Rotire la stînga cu k poziții**

```
For j:=1 to k step 1
    X:=A[N1]
    For I:= N1 to N2-1 step 1
        A[I] := A[I+1]
    End
    A[N2] :=X
End
```

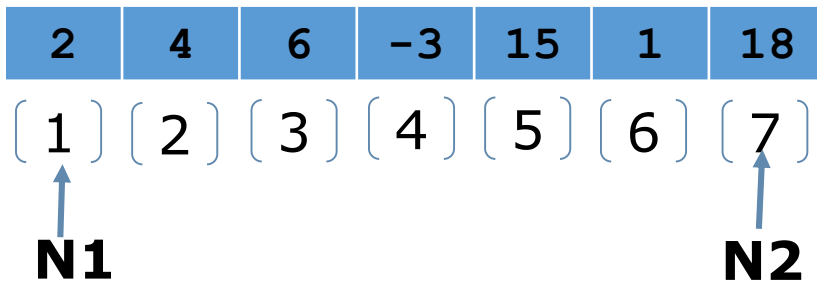
# Rotire la dreapta

Type

Vector = Array[N1..N2] Of Integer

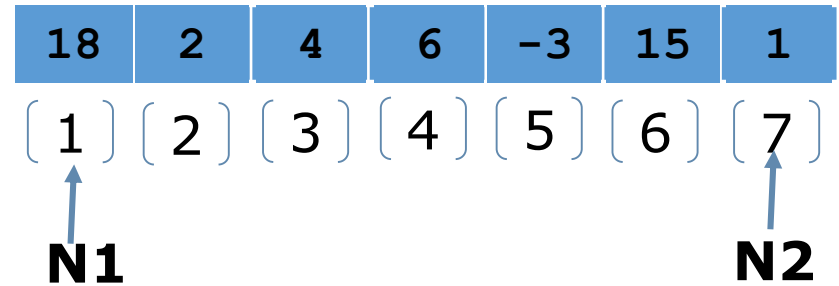
Var

A: Vector; I: Natural



**Rotire la dreapta cu o poziție**

```
X:=A[N2]
For I:= N2 to N1+1 step -1
    A[I] := A[I-1]
End
A[N1] :=X
```



**Rotire la dreapta cu k poziții**

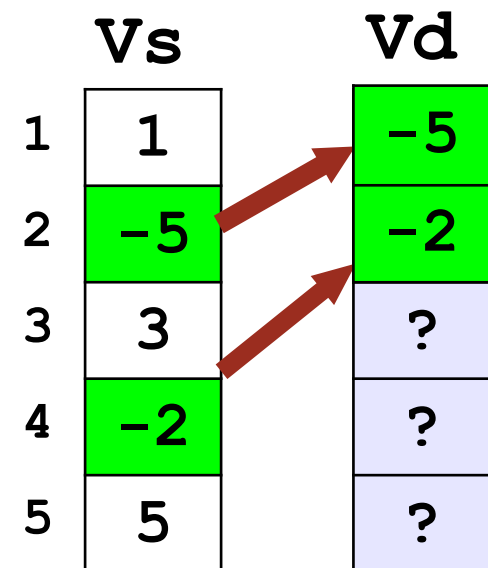
```
For j:=1 to k step 1
    X:=A[N2]
    For I:= N2 to N1+1 step -1
        A[I] := A[I-1]
    End
    A[N1] :=X
End
```

# Copierea valorilor dintr-un vector sursă într-un vector destinație

**Problemă:** elementele din vectorul **Vs**, care satisfac anumitor condiții (de exemplu, cele negative) să fie copiate într-un alt vector.

**Rezolvare:** se va utiliza un contor a elementelor găsite **count**, elementul nou găsit va fi copiat în **Vd[count]**.

```
Const N = 5
Var count, i: natural
    Vs, Vd: array[1..N] of integer
...
count:=0;
for i:=1 to N step 1
    if (Vs[i] < 0) then
        count:=count+1
        Vd[count] := Vs[i]
    end
end
```

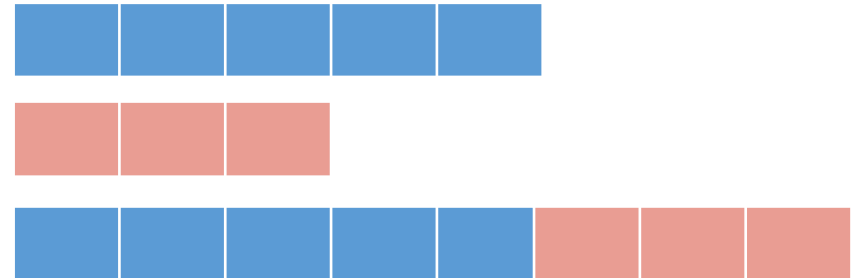


# Metode de formare a vectorului din alți vectori

**Problemă:** Fie vectorul **A** cu  $n$  elemente și vectorul **B** cu  $m$  elemente. Să se formeze vectorul **C** cu  $(n+m)$  elemente.

**Rezolvare:** Primele  $n$  elemente ale vectorului **C** sunt luate din elementele corespunzătoare ale vectorului **A**, iar următoarele  $m$  elemente sunt luate din vectorul **B**.

A: Array[1..N] Of Integer  
B: Array[1..M] Of Integer  
C: Array[1..N+M] Of Integer



```
For I:=1 To N Step 1
    C[I]:=A[I]
End
```

```
For I:=1 To M Step 1
    C[N+I]:=B[I]
End
```



# Metode de formare a vectorului din alți vectori

**Problemă:** Fie vectorul **A** cu  $n$  elemente și vectorul **B** cu  $m$  elemente. Să se formeze vectorul **C**. În **C** se trec toate elementele din **A**, iar din **B** – numai cele care nu sunt în **A**.

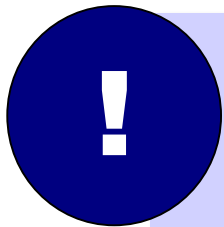
**Rezolvare:** 1) Copierea vectorului **A** în **C**; 2) Copierea în **C** a acelor elemente din **B** care nu sunt în **A**.

```
A: Array[1..N] Of Integer  
B: Array[1..M] Of Integer  
C: Array[1..N+M] Of Integer
```

```
For I:=1 To N Step 1  
    C[I]:=A[I]  
End
```

```
K:=0  
For J:=1 to M step 1  
    X:=B[j]  
    I:=1  
    While (I<=N) and (A[I]#X) do  
        I:=I+1  
    End  
    If I>N Then  
        K:=K+1  
        C[N+K]:=X  
    End  
End
```

# Sortarea vectorilor



Prin **sortare** se înțelege o operație de aranjare a unei mulțimi de elemente într-o anumită ordine.

Modalitatea de ordonare poate fi:

- **în creștere;**
- **descreștere.**

Elementele vectorului  $A_1, A_2, \dots, A_n$  sunt aranjate în ordine **crescătoare**, dacă între elementele vectorului se respectă relația

$$A_1 \leq A_2 \leq A_3 \leq \dots \leq A_n$$

Elementele vectorului  $A_1, A_2, \dots, A_n$  sunt aranjate în ordine **descrescătoare**, dacă între elementele vectorului se respectă relația

$$A_1 \geq A_2 \geq A_3 \geq \dots \geq A_n$$

# Sortarea prin inserție

---

## Ideea rezolvării:

- considerăm, că elementele precedente

**$A[1], A[2], \dots, A[I-1]$**

au fost în prealabil **aranjate în ordine crescândă** și se cere inserarea elementului  $A[I]$  în locul, care îi revine între elementele sortate anterior;

- în continuare următorul **element  $A[I+1]$  va fi inclus între cele ordonate** în mod analogic ș.a.m.d., până când va fi aranjat tot vectorul.

# Sortarea prin inserție

18	13	22	3	10	15	4	1	9
18	13	22	3	10	15	4	1	9
13	18	22	3	10	15	4	1	9
13	18	22	3	10	15	4	1	9
3	13	18	22	10	15	4	1	9
3	10	13	18	22	15	4	1	9
3	10	13	15	18	22	4	1	9
3	4	10	13	15	18	22	1	9
1	3	4	10	13	15	18	22	9
1	3	4	9	10	13	15	18	22

# Sortarea prin inserție

începem cu al doilea element

```
For i := 2 to N-1 step 1
```

```
  X := A[i]
```

```
  J := I-1
```

```
  While (J >= 1) and (A[J] > X) do
```

```
    A[J+1] := A[J]
```

```
    J := J-1
```

```
  end
```

```
  A[J+1] := X
```

```
End
```

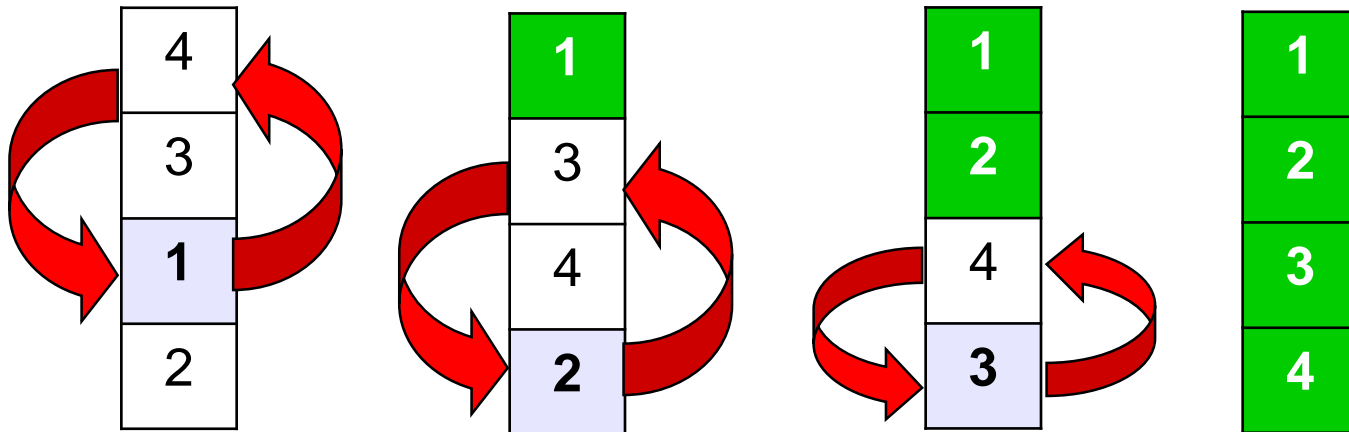
se caută locul lui X  
printre primele (I-1)  
elemente

deplasarea elementelor  
 $A[I-1], A[I-2], \dots, A[J]$  cu o  
poziție la dreapta

# Sortarea prin selecție

## Ideea rezolvării:

- se selectează elementul minim din vector și se interschimbă cu primul element **A[1]**;
- în continuare se caută elementul minim, **începând cu elementul al doilea** al vectorului și se interschimbă cu elementul al doilea al vectorului **A[2]** ș.a.m.d.



# Sortarea prin selecție

18	13	22	3	10	15	4	1	9
18	13	22	3	10	15	4	1	9
1	13	22	3	10	15	4	18	9
1	13	22	3	10	15	4	18	9
1	3	22	13	10	15	4	18	9
1	3	22	13	10	15	4	18	9
1	3	4	13	10	15	22	18	9
1	3	4	13	10	15	22	18	9
1	3	4	9	10	15	22	18	13
1	3	4	9	10	15	22	18	13
1	3	4	9	10	13	22	18	15
1	3	4	9	10	13	15	18	22

# Sortarea prin selecție

vor fi  $N-1$  iterații

```
for i := 1 to N-1 step 1
```

```
  nMin := i
```

```
  for j := i+1 to N step 1
```

```
    if A[j] < A[nMin] then nMin := j end
```

```
  end
```

```
  if nMin <> i then
```

```
    c := A[i]
```

```
    A[i] := A[nMin]
```

```
    A[nMin] := c
```

```
  end
```

```
end
```

determinarea  
elementului minim de la  
 $A[i]$  până la  $A[N]$

dacă este necesar,  
interschimbăm  
elementul minim cu  
elementul  $i$



Putem scăpa de ultimul **if**?

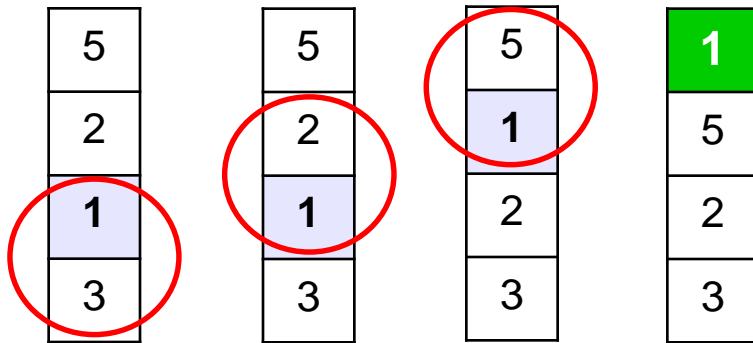


# Sortarea prin interschimbare directă

**Ideea** – bulele de gaz se ridică spre suprafața unui lichid.

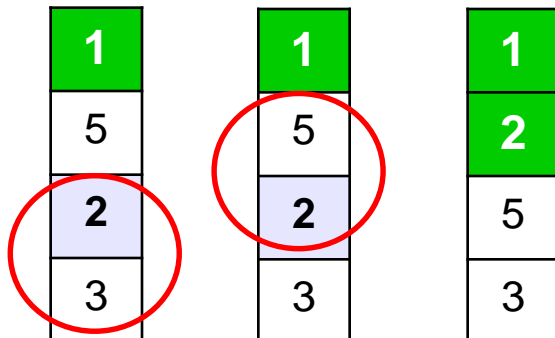
**Pentru vectori** – cel mai mic (elementul “ușor”) se mișcă în sus.

## Prima parcurgere

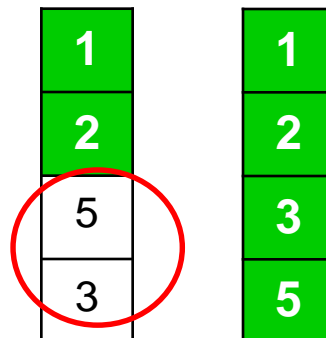


- începând de jos se analizează succesiv perechile de elemente vecine, inversând între ele valorile elementelor care nu îndeplinesc condiția de ordine.
- la o parcurgere a vectorului **un element** (cel mai mic) își ocupă locul său.

## A doua parcurgere



## A treia parcurgere



Pentru sortarea unui vector din  $N$  elemente sunt necesare  $N-1$  parcurgeri (e destul de așezat la locurile sale  $N-1$  elemente).

18	13	22	3	10	15	4	1	9
----	----	----	---	----	----	---	---	---

[illegible]

# Sortarea prin interschimbare directă

```
Const N = 10
Var A: array[1..N] of integer
    i, j, c: integer
Begin
    { completăm vectorul }
    { afișăm vectorul inițial }
    For i:=1 to N-1 step 1
        For j:=N-1 to i step -1
            If A[j] > A[j+1] then
                c := A[j]
                A[j] := A[j+1]
                A[j+1] := c
            end
        end
    end
    { afișăm vectorul ordonat }
End
```

elementele mai sus de  
 $A[i]$  sunt pe locurile sale



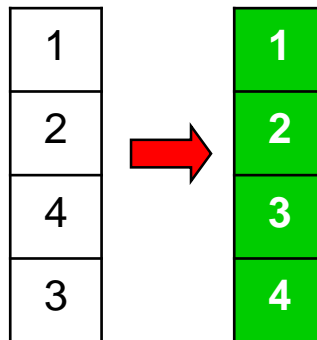
De ce ciclu este de la  $i$   
până la  $N-1$ ?

# Sortarea prin interschimbare directă.

## Algoritmul optimizat

**Ideea** – dacă la executarea metodei bulelor nu au avut loc interschimbări, vectorul este deja ordonat și de alte parcurgeri nu este nevoie.

**Rezolvare:** variabila **Schimb** va arăta au avut loc interschimbări sau nu; în caz că **Schimb** are valoare **False** are loc ieșire.



# Sortarea prin interschimbare directă

```
i := 0
```

```
Repeat
```

```
  i := i + 1
```

```
Var schimb: boolean
```

```
  schimb := False {n-a fost nici o interschimbare}
```

```
  For j:= N-1 to i step -1
```

```
    If A[j] > A[j+1] then
```

```
      c := A[j]
```

```
      A[j] := A[j+1]
```

```
      A[j+1] := c
```

```
      schimb := True {a avut loc interschimbare}
```

```
    end
```

```
  end
```

```
until not schimb { ieșire când schimb = False }
```

# Eliminarea elementului din vector

**Problemă:** Fie vectorul **A** cu  $n$  elemente. Să se elimine din vector toate elemente egale cu  $X$ .

Type

Vector = Array[N1..N2] Of Integer

Var

A: Vector

X, I: Natural

Procedure Delete(K: Natural,

Var X: Vector, Var N:Natural)

Var

I: Natural

Begin

For I:=K to N-1 step 1

X[I] := X[I+1]

End

N := N-1

End

N\_el := N

For I:= N to 1 step -1

If A[I]=X Then

Delete(I, A, N\_el)

End

End

# Includerea elementelor în vector

**Problemă:** Includerea elementului X după elementul cu numărul k.

Type

Vector = Array[N1..N2] Of Integer

Var

A: Vector

X, I: Natural

1. Primele K elemente nu se modifică;
2. Toate elementele din intervalul K+1..N trebuie să fie deplasate la dreapta cu o poziție;
3. Elementului cu numărul K+1 i se atribuie X.

Procedure Insert1 (K, X: Natural,  
Var A: Vector, Var N: Natural)

Var I: Natural

Begin

For I:= N to K+1 step -1

A[I+1] := A[I]

End

A[K+1] := X

N:= N+1

End

# Includerea elementelor în vector

**Problemă:** Includerea elementului X înaintea elementul cu numărul k.

Type

Vector = Array[N1..N2] Of Integer

Var

A: Vector

X, I: Natural

1. Primele K-1 elemente nu se modifică;
2. Toate elementele din intervalul K .. N trebuie să fie deplasate la dreapta cu o poziție;
3. Elementului cu numărul K i se atribuie X.

Procedure Insert2 (K, X: Natural,  
Var A: Vector, Var N: Natural)

Var I: Natural

Begin

For I:= N to K step -1

A[I+1] := A[I]

End

A[K] := X

N:= N+1

End



# Sarcini pentru lucrul independent

---

Fie date următoarele declarații:

**Type**

`Zile = 1..31`

`Consum = Array[Zile] of Real`

**Var**

`Oct: consum`

Elementul `Oct[I]` conține consumul zilnic de energie electrică a unei întreprinderi.

1. Folosind declarațiile să se afișeze zilele cu un consum zilnic maxim.
2. Folosind declarațiile să se determine dacă s-a înregistrat un consum zilnic mai mare de 1000Kwt.