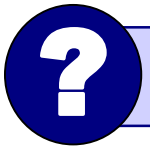
Bazele programării I
Structuri de control. Structura repetitivă

Structura repetitivă

Repetarea unor calcule într-un algoritm se poate face fie multiplicând explicit instrucțiunile respective, fie utilizând construcții speciale care să reprezinte această repetare.

Problemă. Afișați de 5 ori mesajul «Salut».

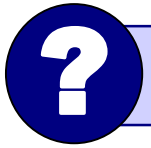
Caracteristica problemei: acțiuni identice au loc de 5 ori.



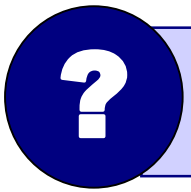
Prin ce metode cunoscute se poate rezolva?

Structura repetetivă

```
Algoritm Mesaj
Begin
 WriteString('Salut')
 WriteString('Salut')
 WriteString('Salut')
 WriteString('Salut')
 WriteString('Salut')
end
```



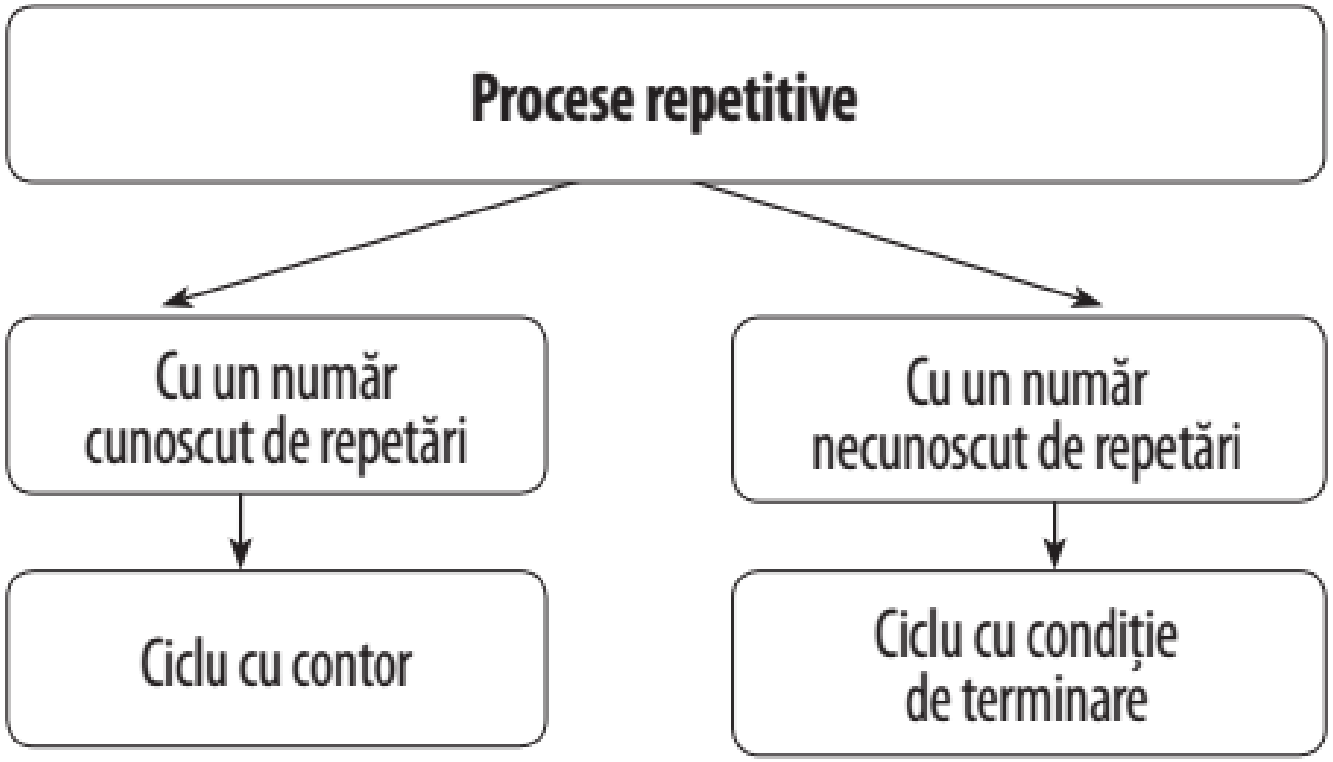
Este oare eficient?



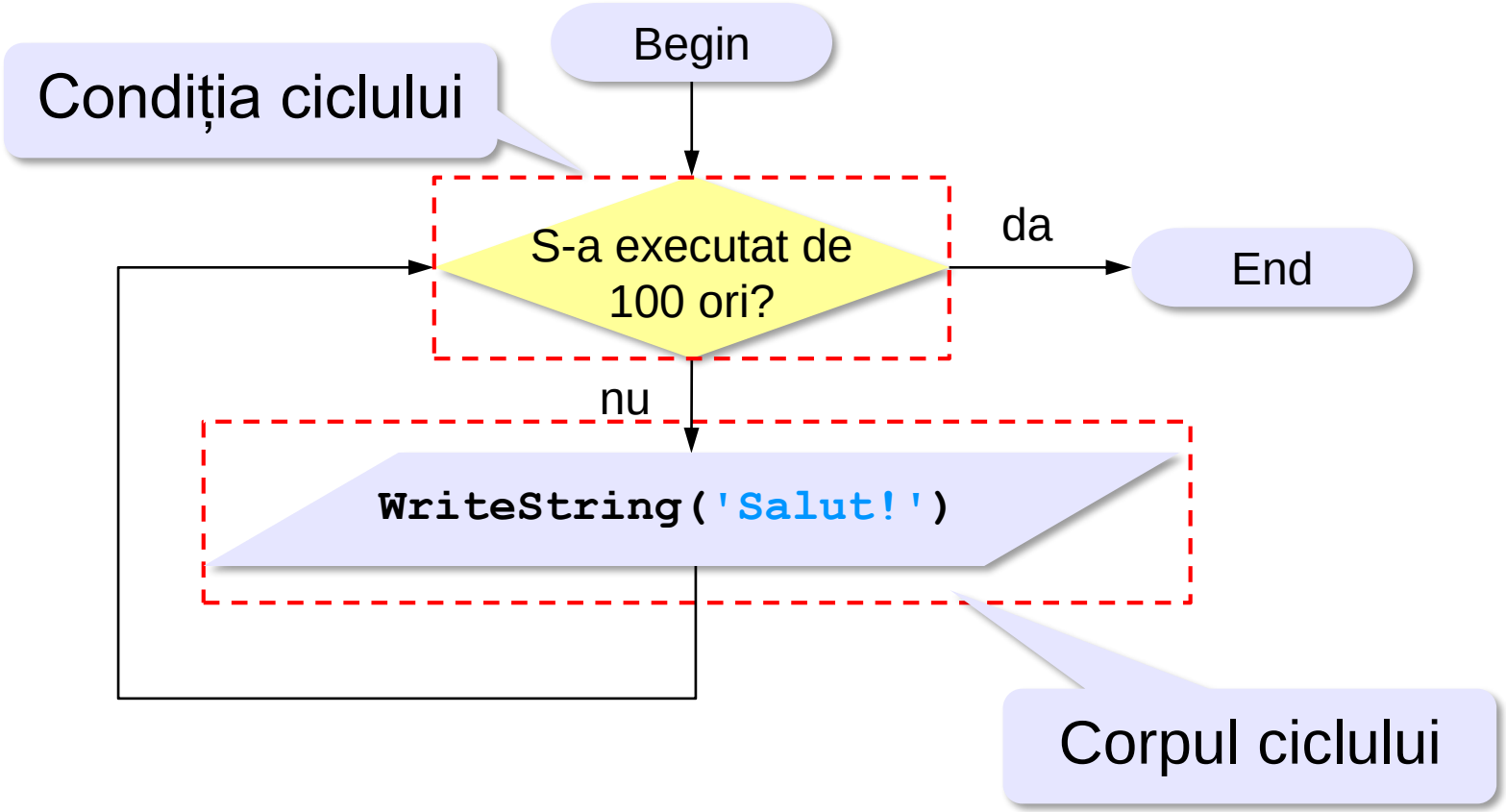
Cum putem însă să procedăm dacă este vorba de 100 de mesaje?

Procese repetitive

Instrucțiunile care descriu execuția repetată a unor prelucrări se numesc **construcții repetitive**.



Elementele structurii repetitive



corpul ciclului indică consecutivitatea de operații, care se execută la o repetare;

condiția ciclului indică de câte ori, cât timp se execută corpul ciclului.

Construcţia While

```
While <condiţie> Do
    …
    … Corpul ciclului
    …
End
```

Condiția – reprezintă o expresie booleană, care exprimă condiția de continuare a ciclului.

Corpul ciclului – reprezintă setul de instrucțiuni care se execută la o repetare.

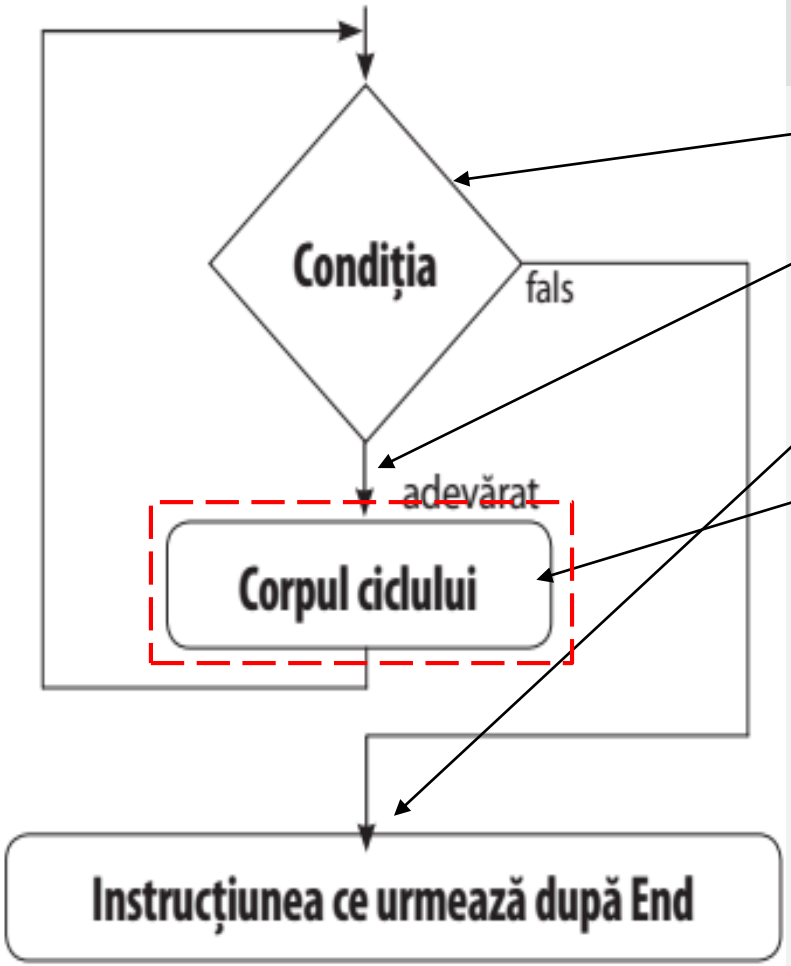
Tehnologia proiectării ciclurilor

1. Se stabilesc comenzile care se realizează la o repetare;

2. Se stabilește condiția ciclului;

3. Se stabilește setul de operații care trebuie să se execute până la începutul ciclului.

Execuţia construcţiei While



Execuția construcției While:

1. Evaluarea condiției:

- *dacă condiția are valoare True, atunci se trece la pasul 2;*
- *dacă condiția are valoarea False, atunci se trece la pasul 4.*

*2. Executarea corpului ciclului (instrucțiunile cuprinse între **Do** și **End**).*

3. Trecere la pasul 1.

*4. Trecere la instrucțiunea ce urmează după **End** (procesul repetitiv se termină).*

Construcţia While

1. Condiția reprezintă **condiția de continuare** a procesului repetitiv.

2. Corpul ciclului se execută atât timp cât valoarea condiției este **True**.

3. Procesul repetitiv se termină atunci când **condiția devine False**.

4. Sunt posibile cazuri când **corpul ciclului nu se execută niciodată**.

Construcția While. Exemplu.

Problemă. Să se determine **numărul cifrelor** în înscrierea zecimală a unui număr întreg pozitiv nenul, păstrat în variabila **n**.

```
contor:= 0
cît timp n > 0
 eliminarea ultimii cifre din n
 mărirea contorului cu 1
```

n	contor
1234	0
123	1
12	2
1	3

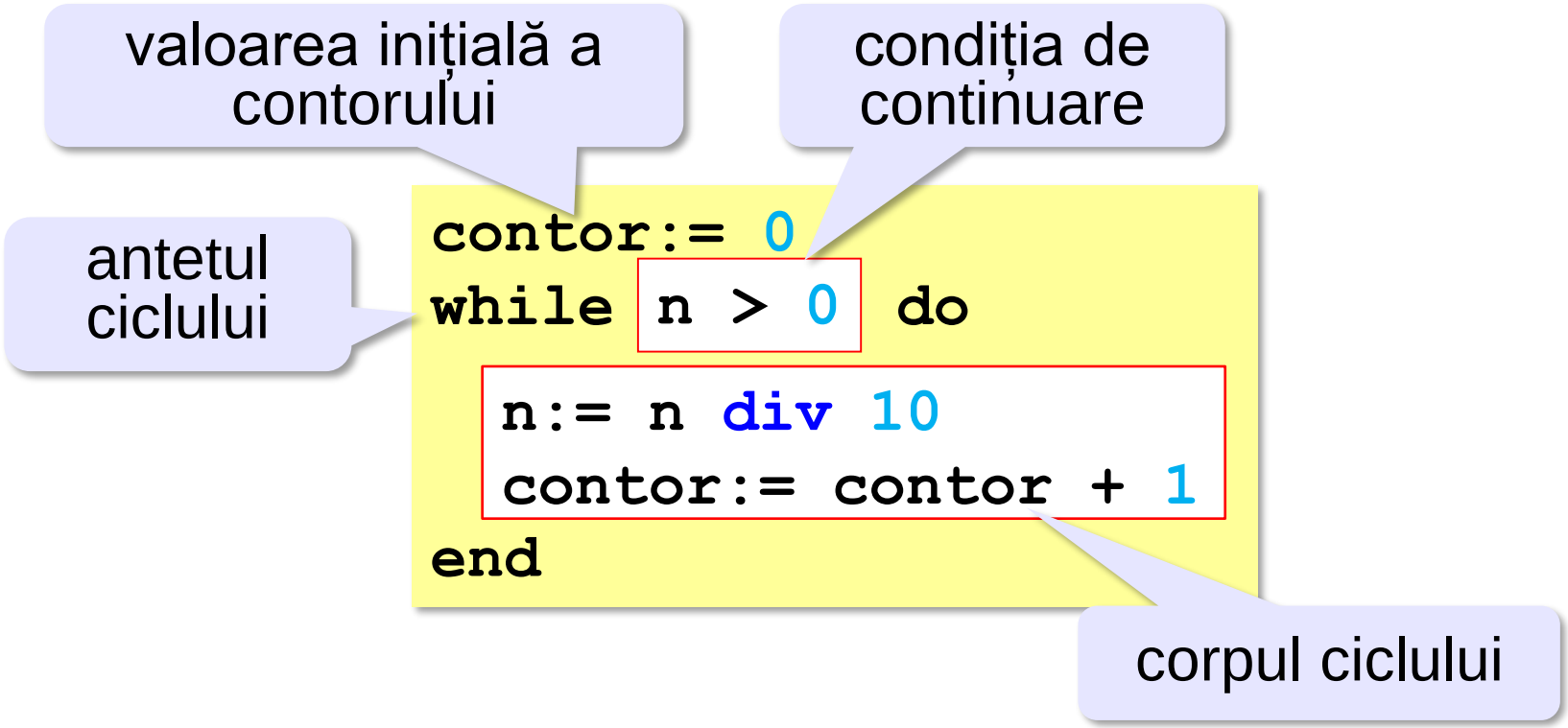
Cum eliminăm ultima cifră?

```
n:= n div 10
```

Cum mărim contorul cu 1?

```
contor:= contor + 1
```

Construcția While. Exemplu.



Ciclu cu test inițial – verificarea are loc la intrarea în ciclu!

De câte ori se repetă corpul ciclului?

```
a:=4; b:=6;
while a < b do a:=a+1;
```

de 2 ori
a = 6

```
a:=4; b:=6;
while a < b do a:=a+b;
```

o dată
a = 10

```
a:=4; b:=6;
while a > b do a:=a+1;
```

niciodată
a = 4

```
a:=4; b:=6;
while a < b do b:=a-b;
```

o dată
b = -2

```
a:=4; b:=6;
while a < b do a:=a-1;
```

la infinit

Organizarea ciclului cu contor

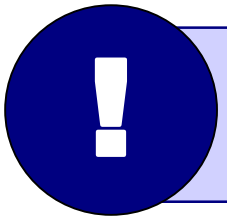
Ciclul cu contor

```
I:=0
While I<N do
    <Operațiile efective ale iterației>
    I:=I+1
End
```

Inițializarea contorului

Verificarea nr. de iterații

Incrementarea nr. de iterații



Dacă variabila ciclului nu se va modifca, atunci ciclul va deveni infnit!

Exemplu. Suma a 4 numere

Să se calculeze suma a 4 numere naturale citite de la tastatură.

```
Var
      I: Natural
      ...
Begin
      I:=0
      ...
      While I<N do
        <operații efective>
         I:= I + 1
      End
End
```

```
N: Integer
S: Integer
```

```
S:= 0
```

```
I < 4
```

Procesul repetitiv se repetă de 4 ori

```
ReadInt(N)
S:= S + N
```

Descrierea algoritmului

```
Algoritm Suma
Var
     S: Natural
     X: Natural
     I: Natural
Begin
     S:=0
     I:=0
     While I<4 do
        ReadNat(X)
        S:=S+X
        I:=I+1
     end
     WriteNat(S)
End
```

Lista variabilelor utilizate

Subproblema 1

Subproblema 2

Subproblema 3

Traseul executării

Instrucțiunea	I	X	S	Condiția I<4	Tastatura	Ecran
	X	X	X			
S:=0			0			
I:=0	0					
While I<4 do				0<4 → True		
ReadNat(X)		3			3	3
S:=S+X			3			
I:=I+1	1					
Evaluarea condiției				1<4 → True		
ReadNat(X)		7			7	7
S:=S+X			10			
I:=I+1	2					
Evaluarea condiției				2<4 → True		
ReadNat(X)		2			2	2
S:=S+X			12			
I:=I+1	3					
Evaluarea condiției				3<4 → True		
ReadNat(X)		5			5	5
S:=S+X			17			
I:=I+1	4					
Evaluarea condiției				4<4 → False		
WriteNat(S)						17

Organizarea ciclului cu condiţie de terminare

Exemplu: de la tastatură se citesc numere întregi până la apariția primului număr negativ.

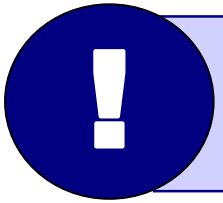
Rezolvare

```
ReadInt(Num)
While Num>=0 do
    <prelucrarea numărului citit>
    ReadInt(Num)
End
```

Citirea primului număr

Ciclul se va repeta atât timp cât numărul nu este negativ

Citirea celorlalte numere



Citirea primului număr se face până la începutul ciclului!

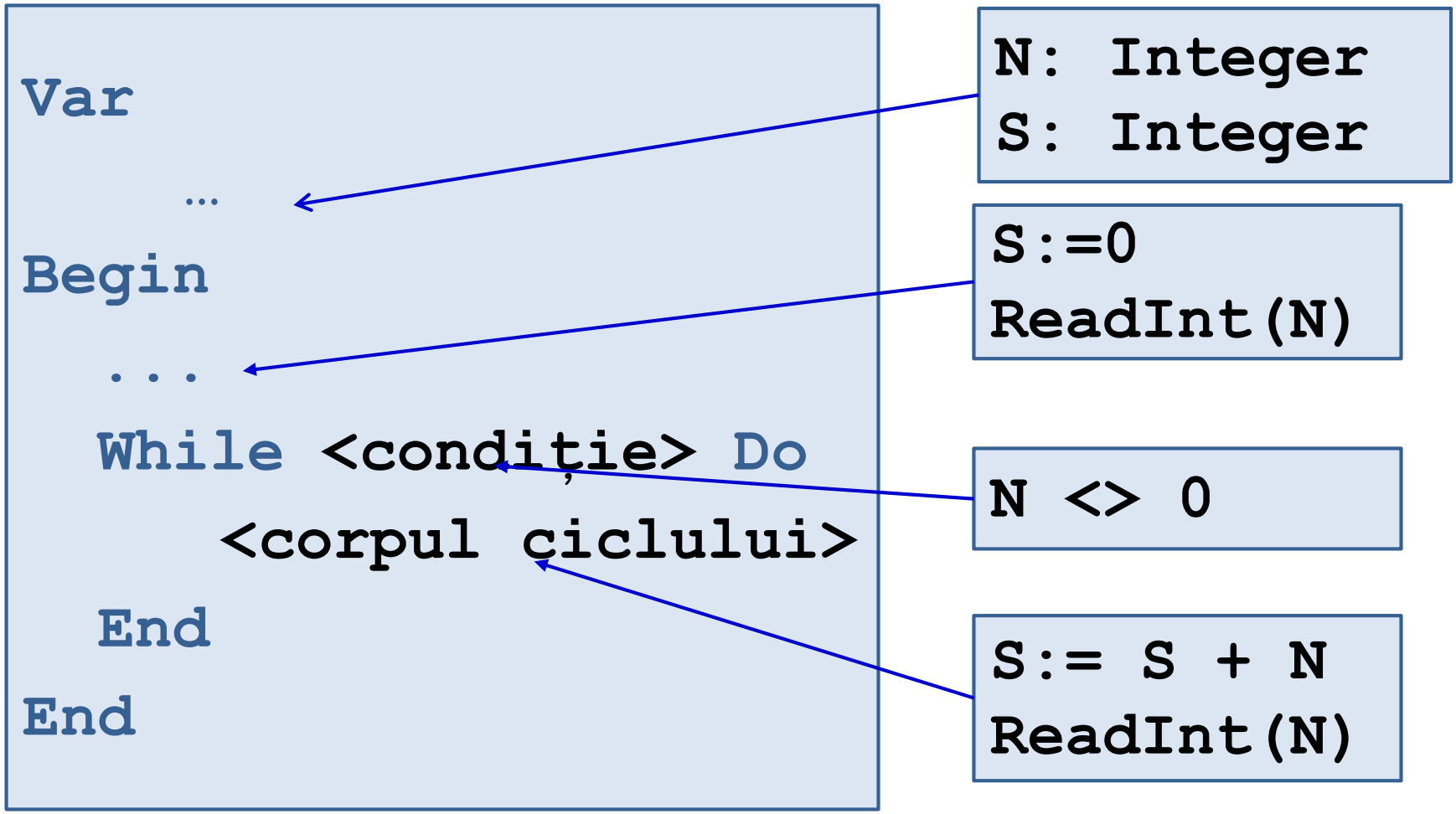
Exemplu

```
Var
     N: Integer
     S: Integer
Begin
     S:=0
     While N <> 0 Do
        ReadInt(N)
        S:= S+N
     End
End
```

N	S	N <> 0
X	X	
	0	nedeterminat

Organizarea ciclului cu condiţie de terminare

Exemplu: De la tastatură se introduce un șir de numere întregi până la primul nul. Să se calculeze suma numerelor introduse.



Exemplu

```
Var
     N: Integer
     S: Integer
Begin
   S:=0
   ReadInt(N)
   While N <> 0 Do
        S:= S + N
        ReadInt(N)
   End
End
```

N	S	N <> 0
X	X	
	0	
2		2 <> 0 -> True
3	2	3 <> 0 -> True
0	5	0 <> 0 -> False

Ciclul cu condiție de terminare

Ciclul cu condiție de terminare în cazul când datele se citesc de la tastatură

```
Var
    X: …
Begin
   Read…(X)
   While  X ≠ <condiţia de terminare a introducerii
                datelor > Do
          <prelucrare X>
          Read…(X)
   End
End
```

Ciclul cu condiţie de terminare

Exemplu: De la tastatură se citesc caractere până la introducerea caracterului ”.”. Să se determine câte cifre au fost introduse.

Algoritmul X

```
Var
    X: Char
    Rez: Natural
Begin
    Rez:=0
    ReadChar(X)
    While X <> '.'  Do
      If (X>='0') and (X<='9') Then
                        Rez:= Rez+1
      End
      ReadChar(X)
    End
    WriteNat(Rez)
End
```

Se citesc caractere. Deci tipul de date va fi Char

Citirea primului caracter

Ciclul se va termina la introducerea “.”

Citirea celorlalte caractere

Exemplu

```
Var
  X: Char
  Rez:  Natural
Begin
  Rez:=0
  ReadChar(X)
  While  X <> '.' Do
    If (X>='0') And (X<='9')Then
      Rez:= Rez + 1
    End
    ReadChar(X)
  End
End
```

X	Rez	X <> '.'
X	X	
	0	
a		a <> '.' -> True
!	0	! <> '.' -> True
5	0	5 <> '.' -> True
.	1	. <> '.' -> False

Practica programării

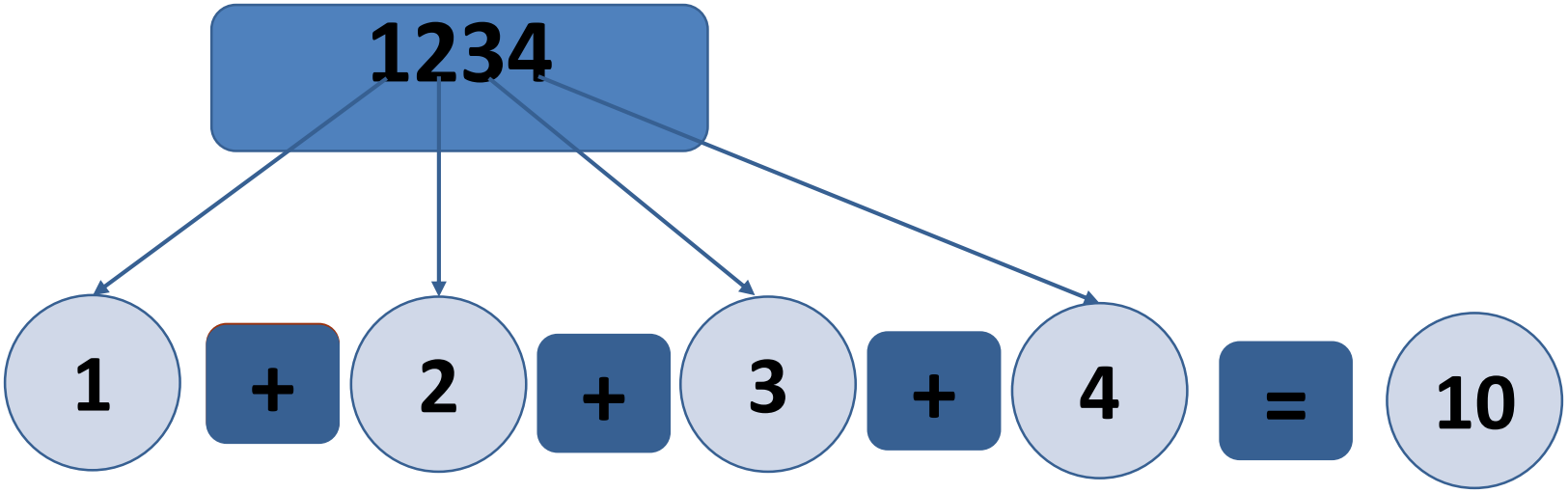
1. Ce se face la o iteraţie. Setul de instrucţiuni care realizează iteraţia va forma corpul ciclului;

2. În ce condiții iterațiile se vor executa; condițiile vor forma condiția ciclului;

3. Ce trebuie de făcut ca prima iterație să se execute corect. Va fi necesar ca aceste instrucțiuni să fie programate până la intrare în ciclu.

Exemplu. Suma cifrelor unui număr

De la tastatură se introduce un număr natural. Să se calculeze suma cifrelor acestui număr. De exemplu, dacă numărul este 1234, atunci suma cifrelor lui va fi 10 (1+2+3+4)



Subproblema 1

Introducerea numărului natural de la tastatură.

```
Var
   N:Natural
```

```
ReadNat(N)
```

1234

1234 mod 10 = 4

1234 div 10 = 123

123 mod 10 = 3

123 div 10 = 12

12 mod 10 = 2

12 div 10 = 1

1 mod 10 = 1

1 div 10 = 0

Subproblema 2

Calcularea rezultatului – suma cifrelor numărului introdus.

```
Var
   S: Natural
   cifra: Natural
```

```
S:= 0
While N>0 Do
      cifra:= N mod 10
      S:= S + cifra
      N:= N div 10
End
```

Subproblema 3

Afișarea rezultatului.

```
WriteNat(N)
```

Exemplu. Suma cifrelor unui număr

```
Algoritm Suma_cifr
Var
      N: Natural
      S: Natural
      cifra: Natural
Begin
   ReadNat(N)
   S:= 0
   While N>0 Do
    cifra:= N mod 10
    S:= S + cifra
    N:= N div 10
   End
   WriteNat(S)
End
```

N	S	Cifra	N > 0
X	X	X	
1234	0		1234>0 ->True
123	4	4	
12	7	3	123>0 -> True
1	9	2	12>0 -> True
0	10	1	1>0 -> True 0>0 -> False

Exemplu. Cel mai mare divizor comun

Să se calculeze cel mai mare divizor comun a două numere naturale m şi N.

Subproblemele:

1. Introducerea a două numere naturale;
2. Determinarea rezultatului – cel mai mare divizor comun a două numere;
3. Afişarea celui mai mare divizor comun a două numere naturale.

Exemplu. Cel mai mare divizor comun

```
Algoritm Cmmdc
    Var
        A, B, Cmmdc: Natural
    Begin
        ReadNat(A)
        ReadNat(B)
        While A<>B Do
            If A>B Then A:=A-B
                   Else B:=B-A
            End
        End
        Cmmdc:=A
        WriteNat(Cmmdc)
    End
```

Lista variabilelor utilizate

Subproblema 1

Subproblema 2

Subproblema 3

Traseul executării

Instrucțiunea	A	B	Cmmd	Condiția A <>B	A>B	Tastatura	Ecran
	x	x	x				
ReadNat(A)	14					14	14
ReadNat(B)		35				35	35
Evaluarea condiției				14 <> 35 True			
Evaluarea condiției					14 > 35 False		
B:=B -A		21					
Evaluarea condiției				14 <> 21 True			
Evaluarea condiției					14 >21 False		
B:= B - A		7					
Evaluarea condiției				14 <> 7 True			
Evaluarea condiției					14 > 4 True		
A:= A -B	7						
Evaluarea condiției				7 <> 7 False			
Cmmd:= A			7				
WriteNat(Cmmd)							7

Exemplu. Palindrom

De la tastatură se introduce un număr natural. Să se determine dacă numărul este sau nu palindrom. Un număr este palindrom, dacă el este egal cu inversul său (se citeşte în ambele direcţii la fel).

De exemplu, numărul 121 este palindrom, iar numărul 123 nu este palindrom.

Subproblemele:

1. Introducerea numărului de la tastatură.
2. Calcularea valorii numărului invers.
3. Afişarea rezultatului (este sau nu palindrom).

Descrierea algoritmului

```
Algoritm Palindrom
Var
    N, X, Cifra, NI: Natural
Begin
    ReadNat(N)
    X:= N
    NI:=0
    While N<>0 Do
       Cifra:= N mod 10
       NI:= NI*10 + Cifra
       N:= N div 10
    End
```

Lista variabilelor utilizate

Subproblema 1

Subproblema 2

Descrierea algoritmului

```
    If NI = X Then
        WriteString('Este palindrom')
     Else
        WriteString('Nu este palindrom')
    End
End
```

Subproblema 3

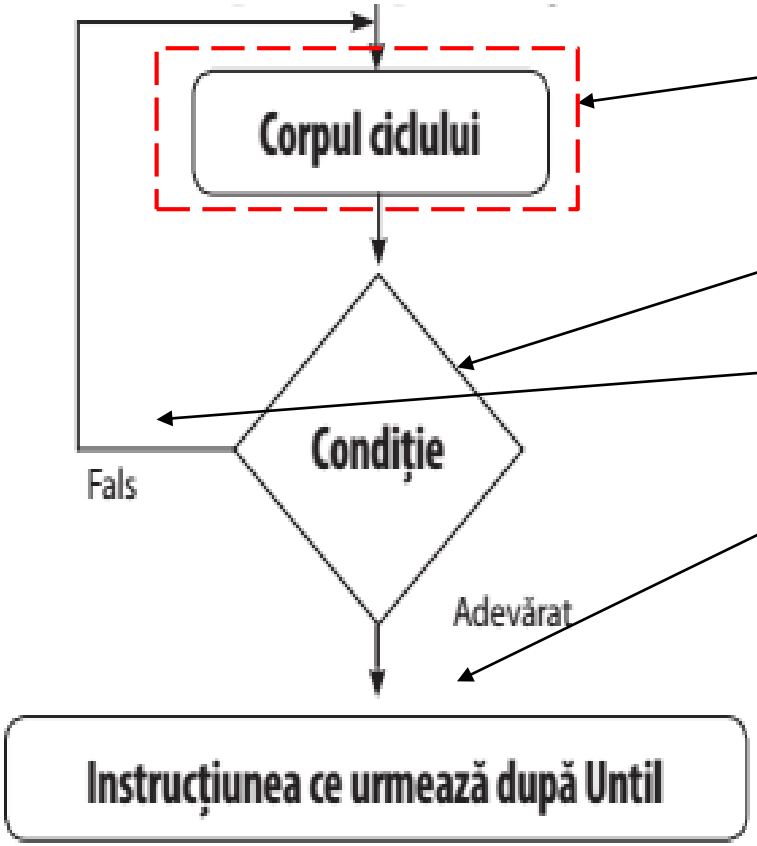
Construcţia Repeat

```
Repeat
    …
    …  Corpul ciclului
    …
Until <condiție>
```

Condiția – reprezintă o expresie logică, care exprimă condiția de finisare a ciclului.

Corpul ciclului – reprezintă setul de instrucțiuni care se execută la o repetare.

Execuţia construcţiei Repeat



Execuția construcției Repeat:

1. ***Executarea corpului ciclului*** *(instrucțiunile cuprinse între Repeat și Until).*
2. ***Evaluarea condiției****:*
   - *dacă condiția are valoare **False**, se revine la pasul 1;*
   - *dacă condiția are valoarea **True**, programul continuă cu instrucţiunea ce urmează după Until, deci se face ieşire din ciclu.*

Construcția Repeat

1. Condiția reprezintă **condiția de terminare** a procesului repetitiv.

2. Corpul ciclului se execută atât timp, cât valoarea condiţiei este **False**.

3. Procesul repetitiv se termină atunci când **condiția devine adevărată**.

4. Corpul ciclului se execută **cel puțin o dată**.

Construcția Repeat

Construcția Repeat este o construcție repetitivă universală.

Poate fi utilizată la exprimarea :

- Proceselor repetitive cu un număr cunoscut de repetări;
- Proceselor repetitive cu un număr necunoscut de repetări.

Tehnologia proiectării ciclurilor

1. Se stabilesc comenzile care se realizează la o repetare;

2. Se stabilește condiția ciclului;

3. Se stabilește setul de operații care trebuie să se execute până la începutul ciclului.

Ciclul cu contor folosind Repeat

Ciclul cu contor

```
I:=0
Repeat
    <Operațiile efective ale iterației>
    I:=I+1
Until I=N
```

Inițializarea contorului

Incrementarea nr. de iterații

Verificarea nr. de iterații

Exemplu. Suma a 4 numere

Să se calculeze suma a 4 numere întregi citite de la tastatură.

```
Var
      I: Natural
      ...
Begin
      I:=0
      ...
      Repeat
        <operații efective>
         I:= I + 1
      Until I = N
End
```

```
X: Integer
S: Integer
```

```
S:= 0
```

```
ReadInt(X)
S:= S + X
```

```
I = 4
```

Procesul repetitiv se repetă de 4 ori

Exemplu. Suma a 4 numere

```
Var
   I: Natural
   X: Integer
   S: Integer
Begin
   I:= 0
   S:= 0
   Repeat
      ReadInt(X)
      S:= S + X
      I:= I + 1
   Until I = 4
   WriteInt(S)
End
```

I	X	S	I = 4
--	--	--	
0			
		0	
	2		
		2	
1			1 = 4 → False
	4		
		6	
2			2 = 4 → False
	1		
		7	
3			3 = 4 → False
	4		
		11	
4			4 = 4 → True

De câte ori se repetă corpul ciclului?

```
a := 4; b := 6;
repeat a := a + 1; until a > b;
```

de 3 ori
a = 7

```
a := 4; b := 6;
repeat a := a + b; until a > b;
```

O dată
a = 10

```
a := 4; b := 6;
repeat a := a + b; until a < b;
```

la infinit

```
a := 4; b := 6;
repeat b := a - b; until a < b;
```

de 2 ori
b = 6

```
a := 4; b := 6;
repeat a := a + 2; until a < b;
```

la infinit

While <expresie logica> **Do**

Corpul ciclului

End

Repeat

Corpul ciclului

Until <expresie logica>

1. Se calculează valoarea de adevăr a **expresiei logice**:
   - Dacă rezultatul **False**, atunci se trece după End;
   - Dacă rezultatul este **True**, atunci:
     - se execută corpul ciclului;
     - se trece la punctul 1.

1. Se execută **corpul ciclului**;
2. Se calculează valoarea de adevăr a **expresiei logice**:
   - Dacă rezultatul **False**, atunci se trece la punctul 1;
   - Dacă rezultatul este **True**, atunci se trece la comanda după Until.

Comparare While - Repeat

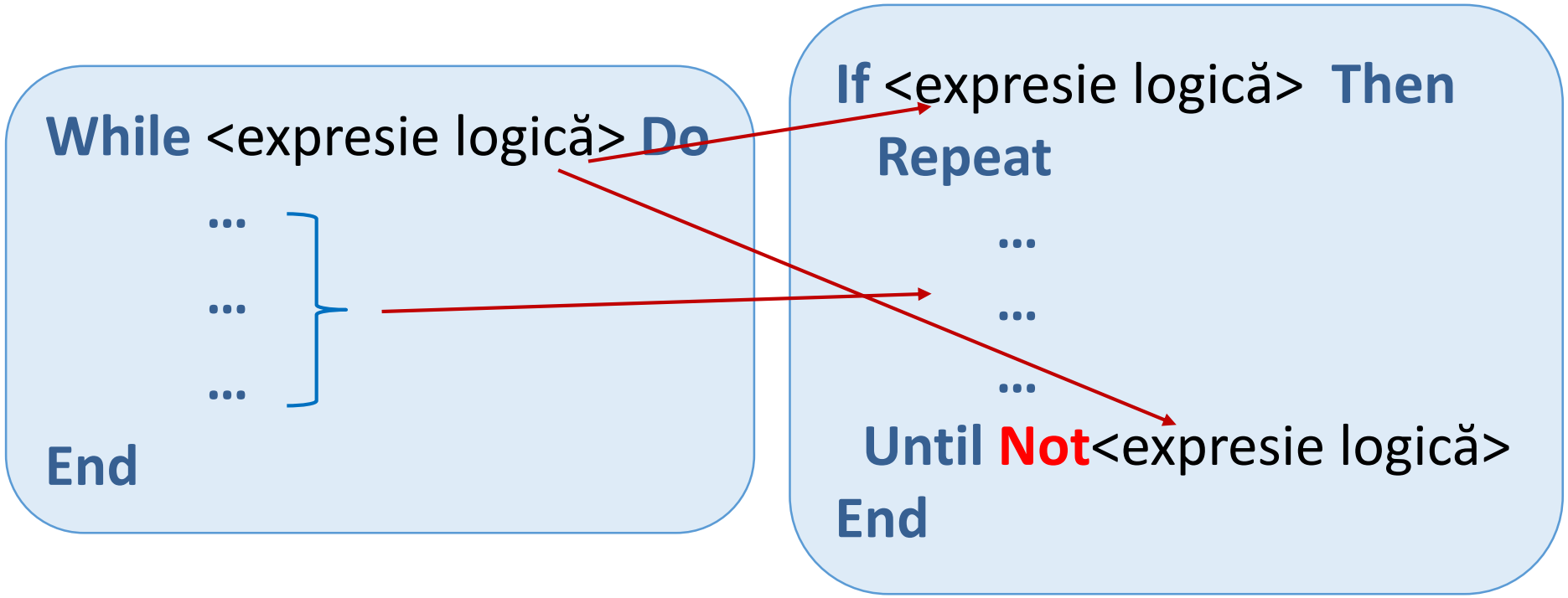
While	Repeat
Expresia logică exprimă condiția de continuare a procesului repetitive.	**Expresia logică** exprimă condiția de terminare a procesului repetitiv.
Corpul ciclului **se execută** atâta timp cât condiția ciclului este adevărată.	Corpul ciclului **se execută** atâta timp cât condiția ciclului este falsă.
Procesul repetitiv **se termină** atunci când expresia logică devine falsă.	Procesul repetitiv **se termină** atunci când expresia logică devine adevărată .
Pot fi cazuri când corpul ciclului nu se va execută nici o dată.	Corpul ciclului se execută cel puțin o dată.

Echivalența construcțiilor Repeat şi While

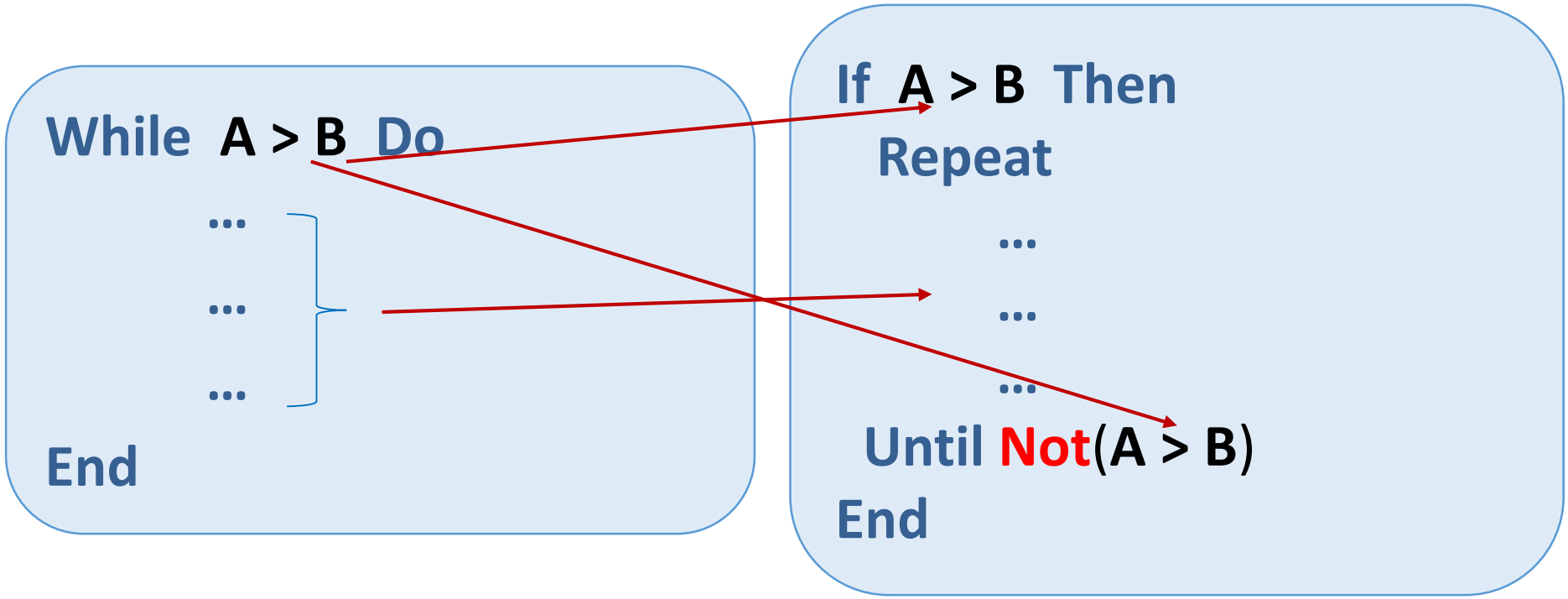
1. Un ciclu While se poate reprezenta cu ajutorul unui ciclu Repeat, dacă acest ciclu se introduce într-o structură de decizie care utilizează condiţia de reluare a ciclului.

2. Un ciclu Repeat se poate înlocui printr-un ciclu While, precedîndu-l pe acesta cu o prelucrare.

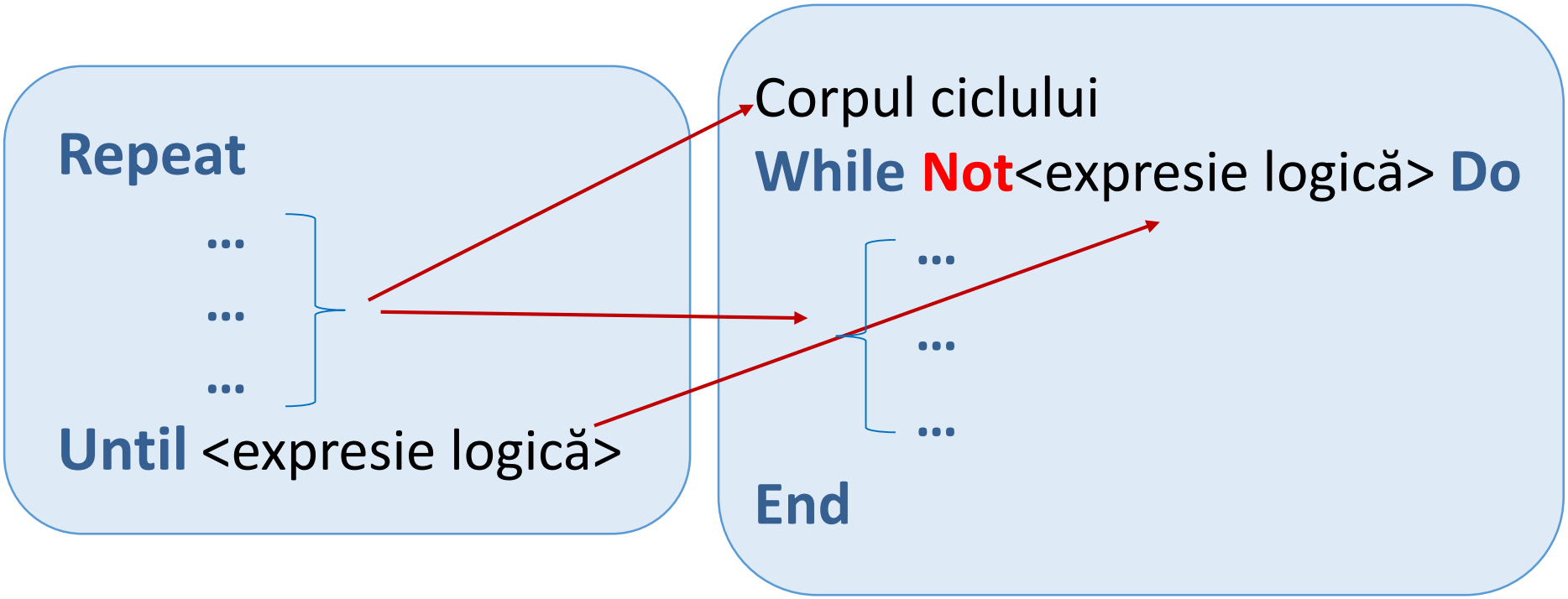
Echivalența While – Repeat



Exemplu While – Repeat



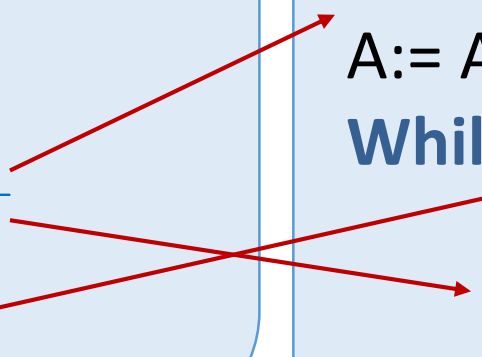
Echivalența Repeat – While



Exemplu Repeat – While

```
B:= 5
A:= 2
X:= 1
Repeat
        X := X + 1
        A:= A + 1
Until A = B
```

```
B:= 5
A:= 2
X:= 1
X := X + 1
A:= A + 1
While Not(A = B) Do
        X := X + 1
        A:= A + 1
End
```



1. Citire A, B (nenule)
2. Calculare produs
3. Afisare produs

```
Var
   A: Natural
   B: Natural
   I: Natural
   P: Natural
```

```
ReadNat(A)
ReadNat(B)
I:=0
P:=0
Repeat
     P:= P+A
     I:= I+1
Until I=B
WriteNat(P)
```

A	B	I	P	I = B
4	3	0	0 4	
		1		1=3 ->False
			8	
		2		2=3 ->False
		3	12	
				3=3-> True

Construcția For

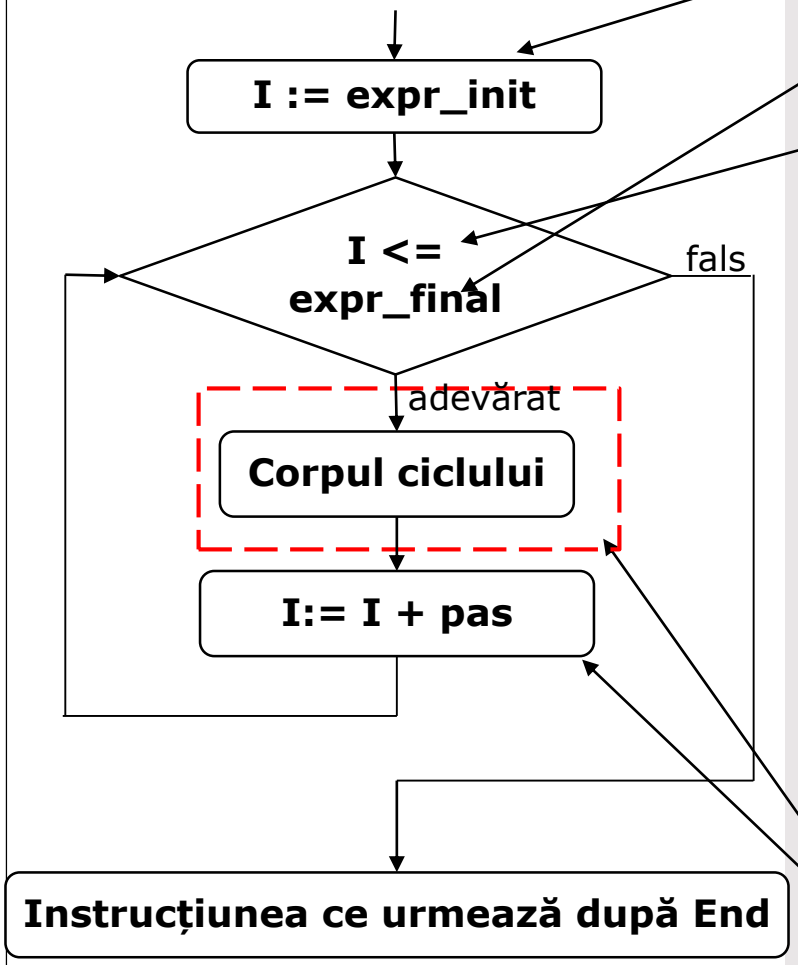
```
For I:= expr_init to expr_final step pas
    …
    …  Corpul ciclului
    …
End
```

I – este un nume de variabilă de tip elementar;

expr_init şi **expr_final** – expresii de același tip cu **I**, numite respectiv expresie inițială şi expresie finală;

pas – poate fi o valoare pozitivă sau negativă.

Funcționarea construcției FOR



1. *Se evaluează valoarea* ***expr_init*** *şi se atribuie valoarea calculată variabilei* ***I****;*

2. *Se evaluează valoarea* ***expr_final****;*

3. *Se compară valoarea variabilei* ***I*** *cu valoarea expresiei* ***expr_final****:*

- *dacă pasul este pozitiv, se evaluează valoarea de adevăr a expresiei* ***I <= expr_final****;*
- *dacă pasul este negativ, se evaluează valoarea de adevăr a expresiei* ***I >= expr_final****.*

4. *Dacă valoarea de adevăr a expresiei evaluate în punctul 3 este* ***True****, se trece la punctul 5, iar dacă valoarea de adevăr este* ***False****, se trece la pasul 8;*

5. *Se execută corpul ciclului;*

6. *Se actualizează valoarea variabilei* ***I:= I + pas****;*

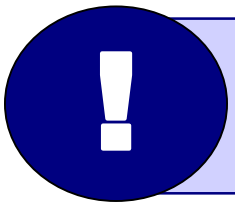
7. *Se trece la pasul 3;*

8. *Gestiunea se transmite la instrucțiunea ce urmează după End*

Organizarea unui ciclu care se repetă de N ori

```
For I:= 1 to N step 1
  ...
  ... Corpul ciclului
  ...
End
```

```
For I:= N to 1 step -1
  ...
  ... Corpul ciclului
  ...
End
```



Corpul ciclului nu trebuie să conţină instrucţiuni care modifcă valoarea variabilei I.

Exemplu. Suma a 4 numere

Să se calculeze suma a 4 numere întregi citite de la tastatură.

```
Var
      I: Natural
      ...
Begin
      ...
 For I:= 1 to N step 1
     <operații efective>
 End
 WriteInt(S)
End
```

```
Num: Integer
S: Integer
```

```
S:= 0
```

```
N = 4
```
Procesul repetitiv se repetă de 4 ori

```
ReadInt(Num)
S:= S + Num
```

Descrierea algoritmului

```
Algoritm  Suma Cu For
Var
    S: Integer
    Num: Integer
    I: Natural
Begin
    S:=0
    For I:= 1 to 4 step 1
        ReadInt(Num)
        S:=S + Num
    End
    WriteInt(S)
End
```

Lista variabilelor utilizate

Subproblema 1

Subproblema 2

Subproblema 3

Traseul executării

Instrucțiunea	I	Num	S	Condiția I<=4	Tastiera	Ecran
	X	X	x			
S:=0			0			
For i:=1 to 4 step 1	1			1<=4 -> True		
ReadINt(Num)		3			3	3
S:= S+Num			3			
I:= I+1	2					
Evaluarea condiției				2<=4 -> True		
ReadINt(Num)		7			7	7
S:= S+Num			10			
I:= I+1	3					
Evaluarea condiției				3<=4 -> True		
ReadINt(Num)		5			5	5
S:= S+Num			15			
I:= I+1	4					
Evaluarea condiției				4<=4 -> True		
ReadINt(Num)		4			4	4
S:= S+Num			19			
I:= I+1	5					
Evaluarea condiției				5<=4 -> False		
WriteNat(S)						19

Exemplu. Funcția $y=x^2$

Să se calculeze valorile funcţiei $y=x^2$ în segmentul [A..B] cu pasul C.

Subprobleme:

1. Introducerea valorilor A, B, C;
2. Afişarea rezultatelor.
    - 2.1 Afişarea antetului;
    - 2.2 Afişarea conţinutului tabelului.

Descrierea algoritmului

```
Algoritm  Functia
Var
    A, B, C: Integer
Begin
    ReadInt(A)
    ReadInt(B)
    ReadInt(C)
    WriteString('_____________')
    Writeln
    WriteString(': X : Y :')
    Writeln
    WriteString('_____________')
    Writeln
```

Lista variabilelor utilizate

Subproblema 1

Subproblema 2.1

Descrierea algoritmului

```
For X:= A To B Step C
    Y:= X*X
    WriteInt(X)
    WriteInt(Y)
    Writeln
End
End
```

Subproblema 2.2

De câte ori se repetă corpul ciclului?

```
a := 1
For i:=1 to 3 step 1
      a := a+1
End
```

de 3 ori
a = 4

```
a := 1
For i:=3 to 1 step 1
      a := a+1
End
```

Niciodată
a = 1

```
a := 1
For i:=1 to 3 step -1
      a := a+1
End
```

Niciodată
a = 1

```
a := 1
For i:=3 to 1 step -1
      a := a+1
End
```

de 3 ori
a = 4

Echivalența construcțiilor For şi While

```
For I:=A to B step C
    <corpul ciclului>
End
```
unde C>0, A<=B

```
I:=A
While I<=B do
    <corpul ciclului>
    I:=I+C
End
```

```
For I:=A to B step C
    <corpul ciclului>
End
```
unde C<0, A>=B

```
I:=A
While I>=B do
    <corpul ciclului>
    I:=I-|C|
End
```

Înlocuirea For cu While și invers

```
for i:=1 to 10 step 1
   {corpul ciclului}
end
```

```
i := 1
while i <= 10 do
   {corpul ciclului}
   i := i + 1
end
```

```
for i:=10 to 1 step -1
   {corpul ciclului}
end
```

```
i := 10
while i >= 1 do
   {corpul ciclului}
   i := i - 1
end
```

Înlocuirea ciclului `for` cu `while` este posibilă întotdeauna.

Înlocuirea `while` cu `for` este posibilă doar atunci, cînd se poate dinainte determina numărul de pași a ciclului.

Echivalența construcțiilor For și Repeat

```
For I:=A to B step C
    <corpul ciclului>
End
unde C>0, A<=B
```

```
I:=A
Repeat
    <corpul ciclului>
    I:=I+C
Until I>B
```

```
For I:=A to B step C
    <corpul ciclului>
End
unde C<0, A>=B
```

```
I:=A
Repeat
    <corpul ciclului>
    I:=I-|C|
Until I<B do
```

Construcţia Loop

Formatul construcției Loop:

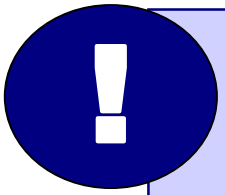
Loop

<corpul ciclului>

End

Execuția construcției Loop:

1. Executarea corpului ciclului;
2. Trecere la pasul 1.



Pentru a termina ciclul Loop, în momentul necesar, se foloseşte operatorul Exit.

```
If <condiție> then
    Exit
End
```

Exemplu. Suma primelor 10 numere

Să se calculeze suma primelor 10 numere naturale.

```
Algoritm Suma_Cu_Loop
Var
    S, X: Natural
Begin
    S:=0
    I:=0
    Loop
      If I>10 Then Exit  End
      S:=S+I
      I:=I+1
    End
    WriteInt(S)
End
```

Cicluri imbricate

Exemplu. Să se afișeze toate numerele prime din diapazonul de la 2 până la 1000.

```
pentru n de la 2 pînă la 1000
  dacă numărul n este prim atunci
    WriteNat(n);
```

nu există divizori [2.. **n-1**]: verificare în ciclu!

? Care număr este «număr prim»?

Cicluri imbricate

```
Algoritm Numere_prime
Var
     n, k, count: Natural
Begin
     for n:=2 to 1000 step 1
        count:=0
        for k:=2 to n-1 step 1
          if n mod k =0 then
            count:=count+1
          end
        end
        if count=0 then
          WriteNat(n)
        end
     end
End
```

Ciclu imbricat

Exemplu. Piramida

Să se realizeze afișarea următoarei piramide:

```
1
1 2
1 2 3
. . . .
1 2 3 … N
```

în care numărul N este introdus de la tastatură.

Descrierea algoritmului

```
Algoritm Piramda
Var
    I, J, N: Natural
Begin
    ReadNat(N)
    For I:= 1 to N step 1
        For j:= 1 to I step 1
            WriteNat(j)
        End
        Writeln
    End
End
```

Afişarea N linii

Afișarea unei linii

Tehnologia proiectării proceselor repetitive

1. Determinarea tipului ciclului;
2. Alegerea celei mai potrivite construcţii repetitive;
3. Determinarea operaţiilor care se execută la o repetare;
4. Determinarea condiţiei ciclului;
5. Determinarea operaţiilor care se vor realiza până la intrarea în ciclu;
6. Verifcarea ciclului proiectat.

Sarcini pentru lucrul independent

1. Să se elaboreze algoritmul care calculează suma S=1+1*2+1*2*3+…+1*2* …*n.
2. Să se elaboreze algoritmul de introducere a unui șir de n numere reale și afișarea valorilor minime și maxime din acest șir, determinate simultan.
3. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere citite consecutiv, cu proprietatea că al doilea număr reprezintă restul împărțirii primului număr la suma cifrelor sale.