**Transformări echivalente între structuri repetitive**

Între anumite tipuri de instrucţiuni repetitive putem avea "rescrieri" perfect echivalente, care pot fi aplicate în orice tip de situaţie.

Le vom prezenta în continuare (sunt 4 tipuri).

Din păcate, orice rescriere care NU se află printre ele, necesită o analiză atentă şi de multe ori se bazează pe un artificiu, formulă de calcul, pe deducerea unui fapt realizat de algoritm, etc.

1) De pe repetitivă cu test iniţial pe repetitivă cu test final

Se dă:

┌cât timp ***condiţie*** execută

│ instrucţiuni

└■

Transformarea sa:

┌dacă ***condiţie*** atunci

│ ┌repetă

│ │ instrucţiuni

│ └până când ***not condiţie***

└■

Obs: În C++ NU există instrucţiune adaptată după "repetă..până când" (spre exemplu,în Pascal există repeat..until, în Basic există do..loop until). Din acest motiv, se acceptă şi scrierea în Pseudocod de tipul execută ... cât timp. Prin urmare, unui programator de C++ îi va veni mult mai uşor să transcrie astfel:

┌dacă ***condiţie*** atunci

│ ┌execută

│ │ instrucţiuni

│ └cât timp ***condiţie***

└■

Exemplu:

Să se rescrie următorul pseudocod utilizând o structură repetitivă cu test final

Iată un exemplu de rulare:

Dacă citim x=4184 se va afişa 4

Dacă citim x=217986 se va afişa 2

Dacă citim x=7 se va afişa 7

Dacă citim x=4 se va afişa 4

citeşte x

pc ← x

┌cât timp pc>9 execută

│ pc ← [pc/10]

└■

scrie pc

Iată rescrierea atât pe "repetă..până când":

citeşte x

pc ← x

┌daca pc>9 atunci

│ ┌repetă

│ │ pc ← [pc/10]

│ └până când pc<=9

└■

scrie pc

... cât şi pe "cât timp..execută":

citeşte x

pc ← x

┌dacă pc>9 atunci

│ ┌execută

│ │ pc ← [pc/10]

│ └cât timp pc>9

└■

scrie pc

Obs: Există o serie de cazuri în care acel "dacă" ce îmbracă repetitiva rescrisă NU mai este necesar. Totuşi, păstrarea sa în scriere, conform "reţetei" de mai sus, NU este greşită, este DOAR inutilă.

2) De pe repetitivă cu test final pe repetitivă cu test iniţial

Se dă:

┌repetă

│ instrucţiuni

└până când ***condiţie***

Transformarea sa:

instrucţiuni

┌cât timp **not *condiţie*** execută

│ instrucţiuni

└■

Obs: Partea mai anevoioasă la acest tip de transformare poate consta în faptul că, dacă instrucţiunile din corpul repetitivei sunt multe, rescrierea lor necesită mai mult spaţiu.

O serie de repetitive pot fi scrise şi mai scurt, însă acest lucru depinde de la caz la caz, neavând o reţetă clară, ci depinde de ingeniozitatea programatorului.

Exemplu:

Să se rescrie următorul pseudocod utilizând o structură repetitivă cu test iniţial:

citeşte n

Iată un exemplu de rulare:

Dacă citim n=3185 se va afişa 4

Dacă citim n=7 se va afişa 1

Dacă citim n=0 se va afişa 1

nc ← 0

┌repetă

│ n ← [n/10]

│ nc ← nc+1

└până când n=0

scrie nc

citeşte n

nc ← 0

n ← [n/10]

nc ← nc+1

┌cât timp n≠0 execută

│ n ← [n/10]

│ nc ← nc+1

└■

scrie nc

3) De pe repetitivă cu contor pe repetitivă cu test iniţial

3b. For descrescător:

┌pentru **contor** ← **li**,**lf,-1** execută

│ **instrucţiuni**

└■

Transformarea sa:

**contor** ← **li**

┌cât timp **contor** >= **lf** execută

│ **instrucţiuni**

│ **contor** ← **contor - 1**

└■

Se dă:

3a. For crescător:

┌pentru **contor** ← **li**,**lf** execută

│ **instrucţiuni**

└■

Transformarea sa:

**contor** ← **li**

┌cât timp **contor** <= **lf** execută

│ **instrucţiuni**

│ **contor** ← **contor + 1**

└■

4) De pe repetitivă cu contor pe repetitivă cu test final

Ne putem gândi că aplicăm 3) şi apoi 1)

3b. For descrescător:

┌pentru **contor** ← **li**,**lf,-1** execută

│ **instrucţiuni**

└■

Transformarea sa:

**contor** ← **li**

┌dacă **contor** >= **lf** atunci

│┌execută

││ **instrucţiuni**

││ **contor** ← **contor - 1**

│└cât timp **contor** >= **lf**

└■

Se dă:

4a. For crescător:

┌pentru **contor** ← **li**,**lf** execută

│ **instrucţiuni**

└■

Transformarea sa:

**contor** ← **li**

┌dacă **contor** <= **lf** atunci

│┌execută

││ **instrucţiuni**

││ **contor** ← **contor + 1**

│└cât timp **contor** <= **lf**

└■