Limbajul Pseudocod

Permite descrierea intr-un limbaj mai putin strict a principalelor structuri de control ale unui algoritm.

1. Citirea

Pseudocod:

citește v1, v2, ...

C++

cin >> v1 >> v2 >> ... ;

2. Afișarea

scrie e1, e2, ...

cout << e1 << e2 << ... ;

3. Atribuire

variabilă ← expr

variabilă = expr;

4. Structura condițională (de decizie)

4A

```
dacă condiție atunci
    secvență
```

```
if(cond)
    secvență;
```

4B

```
dacă condiție atunci
    secv_adevăr
altfel
    secv_fals
```

```
if(cond)
    secv_adevăr;
else
    secv_fals;
```

Ex.:

```
dacă x = 0 atunci
    a ← b+1
    x ← a*a
altfel
    a ← b/x
    x ← a*a
```

→

```
if(x==0)
{   a = b+1;
    x = a*a;
}
else
{   a = b/x;
    x = a*a;
}
```

5) Repetitivă cu test inițial

cât timp condiție execută
    secvență

while (cond)
    secvență

6) Repetitivă cu test final

repetă
    secvență
până când condiție

do
{
    secvență
} while (negare condiție)

7) Repetitiva cu contor

7A)
pentru cnt ← li, lf execută
    secvență

for (cnt = li; cnt <= lf; cnt++)
    secvență;

7B)

pentru cnt ← li, lf, -1 execută
    secvență

Obs: În pseudocod, spre deosebire de C++ NU este permisă modificarea cnt, li sau lf pe secvența din interiorul for-ului.

```
for(cnt=li; cnt>=lf; cnt--)
    secvență;
```

Reguli de transformare între repetitive

∃ 4 reguli fixe, ce pot fi aplicate în orice condiții.

Din păcate, orice regulă care NU face parte dintre acestea trebuie analizată atent și se bazează pe un artificiu, formulă de calcul, caz particular, etc.

1) De pe repetitivă cu test inițial pe repetitivă cu test final:

```
cât timp condiție execută
    secvență
```

⟺

```
dacă condiție atunci
    repetă
        secvență
    până când negare condiție
```

Obs: Dacă doriți puteți scrie și o versiune mai non-standard a lui repetă.. până când și anume

```
execută
    secvență
cât timp condiție.
```

(este acceptată special pentru programatorii de C++.
Evident în acest caz NU veți mai nega condiția!)

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

```
repetă
    secvență
până când condiție
```

⟺

```
Secvență
cât timp nu condiție execută
    secvență
```

3) De pe for pe cât timp:

```
pentru cnt ← li, lf execută
    secvență
```

⟺

```
cnt ← li
cât timp cnt ≤ lf execută
    secvență
    cnt ← cnt + 1
```

4) De pe for pe repetă ... până când

```
pentru cnt ← li, lf execută
    secvență
```

⟺

```
cnt ← li
dacă cnt ≤ lf execută
    execută
        secvență
        cnt ← cnt + 1
    cât timp cnt ≤ lf
```

```
citește n (număr natural)
z←0
p←1
┌cât timp n>0 execută
| c←n%10
| n←[n/10]
|┌dacă c%3=0 atunci
||   z←z+p*(9-c)
||   p←p*10
|└■
└■
scrie z
```

a) n = 1 0 3 4 5 6

răsp:

Urmărim evoluția variabilelor cu tabel.

Ne dăm seama că alg. formează un număr plecând de la cifrele divizibile cu 3 ale celui dat, scăzute din 9 și păstrând ordinea.