Az első tömbös program

Tartalom

Figyeljük meg, miként készül el az informális feladatszövegből először a formális leírása, majd a megoldó program algoritmusa és (Pascal) kódja! Vegyük észre, hogy az egyes lépésekben felhasználjuk az előző lépés(ek)ben már rögzített döntéseinket!

1. Az első feladat

Egy növekvően rendezett sorozat elemeit rendezzük át csökkenőre!
A specifikációt olvasgatva próbáljuk megfogalmazni saját szavainkkal, mit akarnak kifejezni az egyes részek! Pl. N=Hossz(X) jelentése: „X pontosan N elemű sorozat”.

1.1. Specifikáció

Be: $N \in \mathbb{N}, X \in \mathbb{Z}^*$ [helyes így is, csak nem szeretem☺ Be_2: $N \in \mathbb{N}, X \in \mathbb{Z}^N$]

Ki: $M \in \mathbb{N}, Y \in \mathbb{Z}^*$ [helyes így is: Ki_2:$M \in \mathbb{N}, Y \in \mathbb{Z}^M$, vagy csupán így Ki_3: $Y \in \mathbb{Z}^N$]

Ef: $N=Hossz(X) \wedge$ NövekvőenRendezett(X) [a Be_2-höz csak ennyi: NövekvőenRendezett(X)]

Uf: $M=N \wedge \forall i \in [1..N]: X_i \in Y \wedge$ CsökkenőenRendezett(Y)
[a Ki_3-hoz ennyi: $\forall i \in [1..N]: X_i \in Y \wedge$ CsökkenőenRendezett(Y)]

Def: NövekvőenRendezett: $\mathbb{Z}^* \rightarrow \mathbb{L}$
NövekvőenRendezett(x):=$\forall i \in [1..Hossz(x)): x_i \leq x_{i+1}$
CsökkenőenRendezett: … hasonlóan …

1.2. Algoritmus

Az algoritmust felülről lefelé tervezéssel végezzük! Kezdjük a legfelsőbb szintű, ún. „főprogrammal” (és a specifikációból közvetlenül származtatható, ún. globális adatokkal), ezután finomítsuk a felhasznált, de még nem részletezett tevékenységeket (eljárásokat, függvényeket)! Érdemes összeszedegetni, hogy az algoritmus mely részén a specifikáció mely kijelentését használjuk fel közvetlenül/közvetve.
(Gyakorlásképpen a specifikációváltozatok algoritmus-megfelelőit készítsük el!)

```
Program TömbösProgram:
  [a globális adatok:]
  Konstans
    MaxN:Egész(100) [100 helyett bármilyen pozitív egész szám lehetne]
  Típus
    TEgészek=Tömb(1..MaxN:Egész) [hogy majd a paraméterezésnél használhassuk]
  Változó
    N,M:Egész
    X,Y:TEgészek
  [a főprogram:]
  Beolvas(N,X)
  Megfordít(N,X,M,Y)
  Kiír(M,Y)
Program vége.
```

```
[a főprogram finomításai:]

Eljárás Beolvas(Változó n:Egész, x:TEgészek):[1]
  Változó
    i:Egész [a tömb-beolvasáshoz ciklusváltozó]
  Be: n [n∈[0..MaxN], hiszen a TEgészek tömb aktuális max.indexe]
  Be: x(i:1..n) [x(i)≤x(i+1) i∈[1..n)]
Eljárás vége.

Eljárás Megfordít(Konstans n:Egész, x:TEgészek, Változó m:Egész, y:TEgészek):
  Változó
    i[ciklusváltozó]:Egész
  m:=n [az Uf 1. relációja]
  Ciklus i=1-től n-ig [az Uf 2. & 3. relációja]
    y(i):=x(n-i+1)
  Ciklus vége
Eljárás vége.

Eljárás Kiír(Konstans m:Egész, y:TEgészek):
  Változó
    i[ciklusváltozó]:Egész
  Ki: y(i:1..m)
Eljárás vége.
```

1.3. Kód

A kódolás során felvetődő, algoritmusban nem szereplő „újdonságokat” pirossal jelöltük. Bár a kódon nem látszik, de tudjunk róla, hogy célszerű a kódolást is felülről lefelé sorrendben haladva végezni. A kódolás sorrendjénél ez azt jelenti, hogy először a programkód elejét gépeljük be az első alprogramig (**Procedure** UjLap), majd a főprogramot; s ezután kimentjük a kódot. Ezt követően az alprogramok definíciói jönnek, még üres törzzsel; s megint mentünk. S a végén a még hiányzó törzseket is bebillentyűzzük. Alább a kész, teljes kódot látjuk.

```
Program TombosProgram;
(*
  Feladat: Egy növekvően rendezett sorozat elemeit rendezzük át csökkenőre!
  Be: N ELEME Egész, X ELEME Sorozat(Egész)
  Ki: M ELEME Egész, Y ELEME Sorozat(Egész)
  Ef: N=Hossz(X)  ÉS  NövekvőenRendezett(X)
  Uf: M=N  ÉS
      BÁRMELY X-elem ELEME Y  ÉS
      CsökkenőenRendezett(X)
*)
  Uses
    Crt;
  Const
    cim='Megfordítás';
    MaxN=100; {figyelem: itt nincs típusmegadás, sőt hibás lenne!}
  Type
    TEgeszek=Array [1.. MaxN] of Integer;
```

[1] Vegyük észre, hogy a formális paraméterlistán kisbetűket használtam. Ezzel is sugallni szeretném, hogy ezeknek a paramétereknek nem szükségképpen kell a híváskori nevükkel megegyezniük, sőt van úgy, hogy nem is tudnak! Mikor?

```
Var
  N,M:Integer;
  X,Y:TEgeszek;

Procedure UjLap(Const cim:String);
Begin
  ClrScr; Writeln(cim:(80-Length(cim)) Div 2);
End;

Procedure Beolvas(Var n:Integer; Var x:TEgeszek);
  Var
    i{ciklusváltozó}:Integer;
Begin
  Repeat
    Write('Az elemek száma (1..',MaxN,'):');
    Readln(n); {pl. 10}
  Until (0<=n) and (n<=MaxN);
  If n>0 then {legalább egy elem van}
  Begin
    Writeln('Kérem a sorozat elemeit (külön sorban)!');
    Write('1. :'); Readln(x[1]);
    For i:=2 to n do
    Begin
      Repeat
        Write(i,'. :'); Readln(x[i]);
      Until x[i-1]<=x[i]; {monoton növekvő-e?}
    End;{For}
  End;{If n>0}
End;{Beolvas}
```

Az eddigi sok piros jelzi, hogy az algoritmushoz képest sok-sok kódolási döntést kellett hoznunk. Nem így az alábbi, lényegi eljárásnál, amelyet lényegi voltára tekintettel részletesen algoritmizáltunk.

```
Procedure Megfordit(Const n:Integer; Const x:TEgeszek;
                    Var m:Integer; Var y:TEgeszek);
  Var
    i{ciklusváltozó}:Integer;
Begin
  m:=n;
  For i:=1 to n do
  Begin
    y[i]:=x[n-i+1];
  End;
End;{Megfordít}

Procedure Kiir(Const m:Integer; Const y:TEgeszek);
  Var
    i{ciklusváltozó}:Integer;
Begin
  For i:=1 to m do
  Begin
    Writeln(y[i]);
  End;
End;{Kiir}
```

```
Begin {a főprogram:}
  UjLap(cim);
  Beolvas(N,X);
  Megfordit(N,X,MY);
  Kiir(M,Y);
  ReadKey;
End.
```

2. Második feladat

Gondoljuk meg, hogyan módosul a megoldás, ha *helyben* kell átrendezni a sorozatot!

2.1. Specifikáció

Az első probléma az, hogy miként jelöljük ugyanannak az adatnak a bementi és kimenti értékét. Egy lehetőség az, hogy a kimenti értéket egy *aposztróf*fal bővített azonosító jelölje.

Be: $N \in \mathbb{N}, X \in \mathbb{Z}^*$
Ki: $X' \in \mathbb{Z}^*$
Ef: $N=Hossz(X) \wedge NövekvőenRendezett(X)$
Uf: $\forall i \in [1..N]: X'_i \in X \wedge CsökkenőenRendezett(X')$
Def: NövekvőenRendezett: …
CsökkenőenRendezett …

2.2. Algoritmus

Az algoritmus minimálisan módosul. A megváltozott részeket –szokás szerint– pirossal emeltem ki.

```
Program TömbösProgram:
  [a globális adatok:]
  Konstans
    MaxN:Egész(100) [100 helyett bármilyen pozitív egész szám lehetne]
  Típus
    TEgészek=Tömb(1..MaxN:Egész) [hogy majd a paraméterezésnél használhassuk]
  Változó
    N,M:Egész
    X,Y:TEgészek
  [a főprogram:]
  Beolvas(N,X)
  Megfordít(N,X,M,Y)
  Kiír(N,X)
Program vége.

[a főprogram finomításai:]

Eljárás Beolvas(Változó n:Egész, x:TEgészek):
… nincs különbség …
Eljárás vége.
```

```
Eljárás Megfordít(Konstans n:Egész, Változó x:TEgészek
                  Változó m:Egész, y:TEgészek):
  Változó
    i:Egész
  m:=n
  Ciklus i=1-től n Div 2-ig
    Csere(x(i),x(n-i+1)) y(i):=x(n-i+1)
  Ciklus vége
Eljárás vége.

Eljárás Kiír(Konstans m:Egész, y:TEgészek):
… nincs különbség …
Eljárás vége.

[a Megfordít finomítása:]

Eljárás Csere(Változó x,y:Egész):
  Változó
    s:Egész
  s:=x; x:=y; y:=s
Eljárás vége.
```

2.3. Kód

A kódolás során egyetlen újdonságot vezetünk be: ellenőrizni fogjuk a beolvasás közben azt is, hogy a felhasználó egyáltalán szintaktikusan helyes adatot írt-e be.

```
Program TombosProgram;
  …
  Var
    N,M:Integer;
    X,Y:TEgeszek;

  Procedure UjLap(Const cim:String);
  …

  Procedure Beolvas(Var n:Integer; Var x:TEgeszek);
    …
  Begin
    Repeat
      Write('Az elemek száma (1..',MaxN,'):');
      {$i-} Readln(n); {$i+}
    Until (IOResult=0) {szintaktikusan helyes?} and
          (0<=n) and (n<=MaxN) {szemantikusan helyes?};
    If n>0 then {legalább egy elem van}
    Begin
      Repeat
        Writeln('Kérem a sorozat elemeit (külön sorban)!');
        Write('1. :'); {$i-} Readln(x[1]); {$i+}
      Until (IOResult=0) {szintaktikusan helyes?};
```

```
      For i:=2 to n do
      Begin
        Repeat
          Write(i,'. :'); {$i-} Readln(x[i]); {$i+}
        Until (IOResult=0) {szintaktikusan helyes?} and
              (x[i-1]<=x[i]); {szemantikusan helyes?}
      End;{For}
    End;{If n>0}
  End;{Beolvas}

  Procedure Csere(Var x,y:Integer);
    Var
      s:Integer;
  Begin
    s:=x; x:=y; y:=s;
  End;{Csere}

  Procedure Megfordit(Const n:Integer; Var x:TEgeszek~~;~~
                      ~~Var m:Integer; Var y:TEgeszek~~);
    Var
      i{ciklusváltozó}:Integer;
  Begin
    ~~m:=n;~~
    For i:=1 to n Div 2 do
    Begin
      Csere(x[i],x[n-i+1]);~~ y[i]:=x[n-i+1];~~
    End;
  End;{Megfordít}

  Procedure Kiir(Const m:Integer; Const y:TEgeszek);
    …
  End;{Kiir}

Begin {a főprogram:}
  …
End.
```