PROGRAMOZÁSMÓDSZERTAN
3. ELŐADÁS'2004
(VÁZLAT)

1. PASCAL KÓDOLÁSI SZABÁLYOK

1.1. Miről is van szó?

- A kódolás minél mechanikusabbá tétele.
- Csak a nyelv meghatározta dolgokra koncentrálás. Gondoljunk a 'Be:… [..]' szerkezetre, amelyet egészen más hozzáállással kell kódolni pl. Turbo Pascal-ban, mint a Delphiben.
- „Csak" a felhasználóval való zavartalan kommunikáció megszervezése.

Megjegyzés: a programok legtöbbjénél ez viszi el a kód 95%-át (bár nem biztos, hogy a kódolási idő 95%-át). Tehát nem arról van szó, hogy bagatellizálni kell e programozási lépést, hanem csak azt, hogy a feladat lényegét nem a kódolás során kell megoldani (hanem előbb).

- A kódolási szabályok az adott programozási nyelven nyugszanak, tehát egyediek.

1.2. A Turbo Pascal kódolási szabályok

Az alábbiakban nem törekszünk teljességre. A lényeg és az „izgalmasabb" részletek bemutatása a cél. S mintául szolgáljanak más programozási nyelvek kódolási szabályainak megtervezéséhez.

Algoritmikus szerkezet	(Turbo) Pascal-szerkezet
Típusdefiníció	

Konstans
  MaxN:Egész(100)
Típus
  TMag=**Tömb**(1..MaxN:TElem)
  TDátum=**Rekord**(év,hó,nap:Egész)
	Const
	  MaxN=100;
	Type
	  TMag=**Array**[1..MaxN] **of** TElem;
	  TDatum=**Record**
	  ev,ho,nap:Word
	  End;

Típusdeklaráció	

Változó
  N,i:Egész
Konstans
  ma:TDátum(év:2001,hó:10,nap:1)
  mag:TMag
	Var
	  N,i:Integer;
	Const
	  ma:TDatum=(ev:2001;ho:10;nap:1);
	  mag:TMag;

Értékadás	
x:=kif	x:=kif;
Alprogramhívás	
Beolvasás(N,mag)	Beolvasas(N,mag);

<table>
<tr><th colspan="2">Adatbeolvasás</th></tr>
<tr><td>Be: N [0≤N≤MaxN]

Be: mag(1..N) [mag(1)>0 és
mag(2..N-1)≥0 és
mag(N)>0]</td><td>Repeat
{$i-}
Write('Mi a szösz:');
Readln(N);
{$i+}
Until (IOResult=0) and
(N>=0) and (N<=MaxN);
Writeln('Mi a szösz tömb ez?');
Repeat
{$i-}
Write(1:3,'. elem:');
Readln(mag[1]);
{$i+}
Until (IOResult=0) and
(mag[1]>0);
For i:=2 to N-1 do
Begin
Repeat
{$i-}
Write(i:3,'. elem:');
Readln(mag[i]);
{$i+}
Until (IOResult=0) and
(mag[i]>=0);
End;{For i}
Repeat
{$i-}
Write(N:3,'. elem:');
Readln(mag[N]);
{$i+}
Until (IOResult=0) and
(mag[N]>0);</td></tr>
<tr><td colspan="2">…</td></tr>
<tr><td>…</td><td>…</td></tr>
</table>

2. SPECIFIKÁCIÓ – ALGORITMUS – KÓD

2.1. Kapcsolatok

[Adatszerkezet ⇒ Alg.TipDef]

Spec.Be + Spec.Ki ⇒ Alg.TipDef ⇒ Kód.TipDef

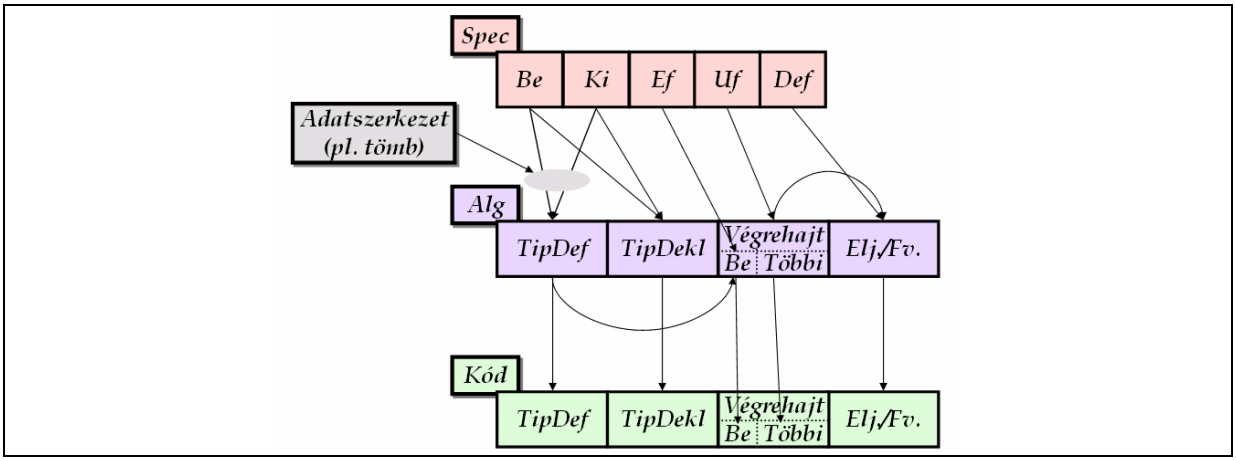
Spec.Be + Spec.Ki + Alg.TipDef ⇒ Alg.TipDekl ⇒ Kód.TipDekl

Spec.Ef ⇒ Alg.Be ⇒ Kód.Beolv

Spec.Uf ⇒ Alg.Végrehajt [⇒ Kód.Elj/Fv[1]] ⇒ Kód.Végrehajt [⇒ Kód.Elj/Fv]

Spec.Def ⇒ Alg.Elj/Fv ⇒ Kód.Elj/Fv

[1] Ti. a finomítások eljárásai, függvényei.



1. ábra. Specifikáció – Algoritmus – Kód.

2.2. Egy korábbi feladatpélda:

Specifikáció:

Be: $N \in \mathbf{N}$, $Hallgatók \in (Név \times Szak \times Évfolyam)^*$,
$Név = \mathbf{S}$, $Szak = \{Info, Mat, Fiz, \ldots^2\}$, $Évfolyam = \mathbf{N}$

Ki: $Vane \in \mathbf{L}$

Ef: $Hossz(Hallgatók) = N \wedge$
$\forall i \in [1..N]: Hallgatók_i.Név \neq '' \wedge Hallgatók_i.Évfolyam \in [1..5]$

Uf: $Vane = \exists i \in [1..N]: Hallgatók_i.Szak = Info \wedge Hallgatók_i.Évfolyam = 1$

Adatszerkezet: Tömb ⇒ Alg.TipDef :

```
Konstans MaxN:Egész(100)
```

Spec.Be + Spec.Ki ⇒ Alg.TipDef :

```
Típus   TNév=Szöveg
        TSzak=(Info, Mat, Fiz, …)
        TÉvfolyam=Egész
        THallgató:Rekord(
                    Név:TNév
                    Szak:TSzak
                    Évfolyam:TÉvfolyam)
        THallgatók=Tömb(1..MaxN:THallgató)
```

Spec.Be + Spec.Ki + Alg.TipDef ⇒ Alg.TipDekl :

```
Változó N:Egész
        Hallgatók:THallgatók
        Vane:Logikai
```

[2] A „…” jelentése: itt egy felsorolás lenne, amit most nem folytatunk) formálisan persze kellene, mivel ki nem található)! Figyelem: így csak egy-szak rendelhető a hallgatóhoz! Nagy baj ez?

Spec.Ef ⇒ Alg.Be ⇒ Kód.Beolv

```
Be: N [0≤N≤MaxN]
Be: Hallgató(1..N)
 [Hallgató(1..N).Név≠'' és
  Hallgató(1..N).Évfolyam∈[1..5]]
```

```
1. előbb
```

Spec.Uf ⇒ Alg.Végrehajt ⇒ Kód.Végrehajt

```
1. korábbi előadáson
```

```
…
```

3. MIT JELENT A TÉTELBIZONYÍTÁS

Figyeljük meg például a Kiválasztás tétel bizonyítását!

Specifikáció:

Be: $N \in \mathbb{N}$, $X \in H^*$, $T: H \to \mathbb{L}$

Ki: $Sorsz \in \mathbb{N}$

Ef: $Hossz(X)=N \wedge \exists i \in [1..N]: T(x_i)$

Uf: $Sorsz \in [1..N] \wedge T(x_{Sorsz})$

Alg:

```
Konstans MaxN:Egész(???)
Típus THk=Tömb(1..MaxN:TH)
Eljárás Kiválasztás(Konstans N:Egész, X:THk
                    Változó  Sorsz:Egész):
  Változó
    i:Egész
  i:=1
  Ciklus amíg nem T(X(i))
    i:+1
  Ciklus vége
  Sorsz:=i
Eljárás vége.
```

3.1. A bizonyítás ötlete

A ***helyes*** program mint ***állapottér-transzformáció*** az ***Ef*** által behatárolt részállapotból indul ki, és utasításról utasításra haladva transzformálja az állapottér egyes komponenseit mindaddig, amíg az ***Uf*** által meghatározott végállapotba nem jut.

Kövessük tehát mi is utasításról utasításra az állapottér transzformált részállapot-halmazait! Vagyis adjuk meg a ***predikátumok****at* az egyes utasítások mögött kiindulva az ***Ef***-ből. Ha eljutottunk az ***Uf***-be, akkor a bizonyítás kész.

3.2. Az „egyenes” bizonyítás lehetetlensége

1. Egy feladat specifikációjában az ***Ef***-t tetszőlegesen ***szűkíthetjük***, a korábbi megoldás megoldás marad ezután is.

   Például, Ef: … $\exists i \in [1..N \text{ Div } 2] : T(x_{2*i})$ – *azaz létezik páros indexű T-tulajdonságú*

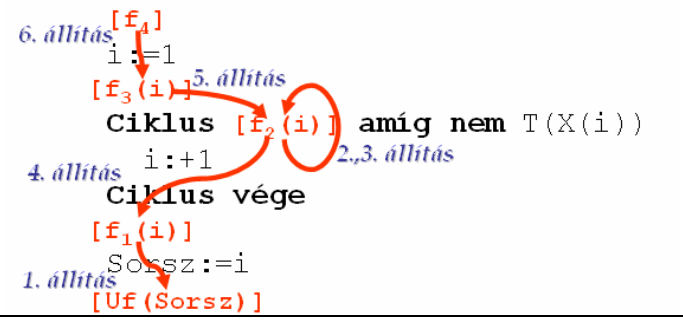
2. Az ***Ef***-nek vajmi kevés ***közölni valója van*** a feladatról. Már csak azért is igaz ez a kijelentés, mert a tételek között soknak ugyanaz (pl. AZONOSAN IGAZ) az Ef-e.

3.3. Kiindulópont

A legtöbb ***támpontot*** a feladatról az ***Uf*** ad. Ebből kell tehát kiindulni, s követni *visszafelé* a „transzformáció-inverzének" a működését, amíg az Ef-hez el nem érünk.
A transzformáció utáni állapotból az utasítás előttire következtetni nem mindig könnyű, ezért *segédállítások*at (és rajtuk nyugvó következtetési szabályokat) kell találni.
Az elméleti részletek mellőzésével élvezzük a levezetés báját, lényegét.

3.4. Menet

Az algoritmus tevékenység részére kell figyelnünk csak. Megfelelő helyekre illesztünk be állításokat. Az állítások paraméterfüggését korlátozzuk a legjellegzetesebb adatra. Pl. az X tömbtől való függést, mint nyilvánvalót, nem jelöljük.



2. ábra. A bizonyítás vázlata.

0.

```
[f4]
 i:=1
[f3(i)]
 Ciklus [f2(i)] amíg nem T(X(i))
   i:+1
 Ciklus vége
[f1(i)]
 Sorsz:=i
[Uf(Sorsz)]
```

Uf(Sorsz): Sorsz∈[1..N] ∧ $T(x_{Sorsz})$

1.

```
[f4]
 i:=1
[f3(i)]
 Ciklus [f2(i)] amíg nem T(X(i))
   i:+1
 Ciklus vége
[f1(i)]
 Sorsz:=i
[Sorsz∈[1..N]  ∧  T(xSorsz)]
```

***1. állítás**: ha Uf(Sorsz) igaz a 'Sorsz:=i' után, akkor előtte igaz kell legyen:*
$f_1(i)$: i∈[1..N] ∧ $T(x_i)$.
Ui.: helyettesítsük be i-t Uf-ben a Sorsz helyére! Mivel Uf(Sorsz) igaz, és a 'Sorsz:=i' után épp i értékét örökli, így nyilvánvaló: Uf(i)=Igaz. Azaz helyes:
$f_1(i)$◄Sorsz:=i►Uf(Sorsz)[3] [4] ◊

[3] A „p_1◄u►p_2" jelentése: „a p_1–ből az u utasítás végrehajtása után igaz p_2".
[4] Általában is igaz, hogy „**P(f(x))◄y:=f(x)►P(y)**". Ezt hívják Hoare rendszerében értékadási axiómának.