Az összetett programozási tételek is egy tőről fakadnak

Zsakó László[1], Törley Gábor[2], Szlávi Péter[3]

[1]zsako@caesar.elte.hu, [2]pezsgo@inf.elte.hu, [3]szlavi@elte.hu
ELTE IK

Absztrakt. A programozás tanulás egyik klasszikus útja a programozási feladatok nagy csoportokba, ún. programozási tételekbe sorolása, majd konkrét feladatok visszavezetése a programozási tételekre. Az egyes tanítási módszerek eltérő számú programozási tételt vezetnek be a tanulás során, majd még kombinálják is őket egymással. Ebben a cikkben megmutatjuk, hogy az összetett programozási tételek ugyanabból a tőről fakadnak, mint az elemiek, azaz elvi szempontból elég lenne egyetlen programozási tételt kimondani, majd mindent arra visszavezetni. A cikk végén visszatérünk arra, hogy a több programozási tételnek mi a gyakorlati értelme.

Kulcsszavak: programozási tétel, programozási módszertan, módszeres programozás, algoritmus, specifikáció

1. Alapvető tudnivalók

A programozási feladatok megoldásához fel kell ismernünk a programozási feladatokban rejlő lényeges jellemzőt: a feladatok nagy feladatosztályokba sorolhatók a feladat jellege szerint, s e feladatosztályokhoz tudunk készíteni a teljes feladatosztályt megoldó algoritmusosztályt. Ezeket a típusfeladatokat programozási tételeknek nevezzük, ugyanis bebizonyítható, hogy a megoldásaik a feladat garantáltan helyes megoldásai. Számuk a tanítási módszertantól függően más-más lehet. [1,2,3,4]

E programozási tételek ismeretében már a legtöbb feladat megoldása során nincs más dolgunk, mint felismerni a megfelelő programozási tételt, megfeleltetni a konkrét adatait az általános feladatosztálynak, majd ezeket az általános megoldó algoritmusban a konkrétumokkal helyettesíteni. Ezzel a módszerrel a feladatnak mindig egy – elvileg – helyes megoldását kapjuk. [5,6]

Ezek a feladatok mindig olyanok lesznek, hogy egy (vagy több) adatsokasághoz kell rendelnünk valamilyen eredményt. E sokaságot az egyszerűség kedvéért mindig valamilyen sorozatként fogjuk fel. A „sorozatság” az elemek feldolgozási sorrendjét határozza meg. A legtöbb esetben elég olyan sorozatokkal foglalkozni, amelyek elemei egymás után – egyesével – feldolgozhatók. Ez a bemenő sorozatra olyan művelet létét feltételezi, amely elölről, egyenként képes adni a sorozat elemeit, az eredmény sorozatba pedig az addig belekerült elemek mögé képes tenni egy új elemet. Egyszerű esetben a sorozatokat mindig egy tömbbel ábrázoljuk.

Az elemi programozási tételekkel egy előző cikkünkben foglalkoztunk [7]. Az összetett programozás tételek sorozat(ok)hoz sorozat(ok)at rendelő feladatok:

- másolás,
- kiválogatás,
- szétválogatás,
- metszet,
- unió.

Az első programozás tétel [7]

Az első programozási tétel egy egyszerű feladatból, számok összegének kiszámításából alakul ki. Az alábbiakban megadjuk a feladat specifikációját és algoritmusát, azaz a programozási tétel két kellékét. Alkalmazzuk [5,6]-beli jelöléseket.

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $\Sigma: H^* \to H$, $+: H \times H \to H$ (ahol $H=\mathbb{N}$ vagy $H=\mathbb{Z}$ vagy $H=\mathbb{R}$)
$\Sigma(X_1,...,X_N) = \Sigma(X_1,...,X_{N-1}) + X_N$, $F() = 0$

Kimenet: $S \in H$

Előfeltétel: –

Utófeltétel: $S = \Sigma(X_1,...,X_N)$

```
Sorozatszámítás(N,X,S):
  S:=0
  Ciklus i=1-től N-ig
    S:=S+X[i]
  Ciklus vége
Eljárás vége.
```

Ezt a feladatot általánosítva kapjuk meg a **sorozatszámítás** programozási tételt. A **+** műveletet általánosítjuk:

1) az általánosított, bináris (**f**) műveletnek legyen (baloldali) nulla eleme ($\mathbf{F_0}$);
2) a műveletek egymás utáni alkalmazásának eredménye pedig ne függjön a végrehajtási sorrendtől, azaz asszociatívak legyenek.

Az **f** bináris műveleten és az $\mathbf{F_0}$ nulla elemen nyugvó, H^*-on értelmezett műveletet **F**-fel jelöljük.

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $F: H^* \to H$, $f: H \times H \to H$, $F_0 \in H$
$F(X_1,...,X_N) = f(F(X_1,...,X_{N-1}),X_N)$, $F() = F_0$

Kimenet: $S \in H$

Előfeltétel: –

Utófeltétel: $S = F(X_1,...,X_N)$

Megjegyzés: F sokszor Σ, átlag, szórás, skalárszorzat, Π, $\cup$, $\cap$, $\forall$, $\exists$, egymás után írás, Max, Min.

```
Sorozatszámítás(N,X,S):
  S:=F0
  Ciklus i=1-től N-ig
    S:=f(S,X[i])
  Ciklus vége
Eljárás vége.
```

Megjegyzés: itt és a továbbiakban **vastagon** szedéssel jelezzük az előző algoritmushoz képesti változtatásokat.

Másolás – függvényszámítás

A sorozatszámítás tételét megfogalmazhatjuk úgy is, hogy a kimeneti tömb (**Y**) úgy épüljön fel, hogy a bemeneti tömb összes elemén alkalmazunk egy függvényt (**g**), és az így kapott elemet a kimeneti tömb végére (**f**) tesszük. Ekkor Y „nulleleme" ($\mathbf{F_0}$) az üres tömb, az **F** függvény gyarapítani fogja **Y** elemeit a $\mathbf{g(X_i)}$ elemeivel. Az **F** függvény gyakorlatilag a tömbre értelmezett összeadás művelete lesz (ezt jelöltük a „végére" művelettel).

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $g: H \to G$, $F: G^N \to G$, $f: G^* \times G \to G$, f – végére

Kimenet: $Y \in G^N$

Előfeltétel: –

Utófeltétel: $\forall i(1 \le i \le N)$: $Y=F(g(X_1),\ldots, g(X_N))$

```
Másolás(N,X,Y):
  Y:=üres
  Ciklus i=1-től N-ig
    Y:=végére(Y,g(X[i]))
  Ciklus vége
Eljárás vége.
```

Ha a bemenet és kimenet tömbök helyett egy-egy szöveg típusú változó lenne, akkor a fenti algoritmus így nézne ki:

```
Másolás(N,X,Y):
  Y:=""
  Ciklus i=1-től N-ig
    Y:=Y+g(X[i])
  Ciklus vége
Eljárás vége.
```

Ez pedig még inkább utal a sorozatszámításra.

Mivel **Y** minden eleme $\mathbf{g(X_i)}$ lesz, a tétel így is megfogalmazható (az előző algoritmus **g** függvényét most **f**-fel jelöltük):

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $f:H \to G$

Kimenet: $Y \in G^N$

Előfeltétel: –

Utófeltétel: $\forall i(1 \le i \le N)$: $Y_i=f(X_i)$

```
Másolás(N,X,Y):
  Ciklus i=1-től N-ig
    Y[i]:=f(X[i])
  Ciklus vége
Eljárás vége.
```

Végül egy egyszerű átalakítással jutunk el a feltételes másolás tételéhez, azaz csak a T tulajdonságú elemekre alkalmazzuk a függvényszámítást.

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $f:H \to G$, $T:H \to \mathbb{L}$

Kimenet: $Y \in G^N$

Előfeltétel: –

Utófeltétel: $\forall i(1 \le i \le N)$ $T(X_i) \to Y_i=f(X_i)$ és nem $T(X_i) \to Y_i=X_i$

```
Másolás(N,X,Y):
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Y[i]:=f(X[i]) különben Y[i]:=X[i]
  Ciklus vége
Eljárás vége.
```

Kiválogatás

Ha a sorozatszámításnál az **F** függvényt a tömbre értelmezett feltételes összeadással helyettesítjük, akkor megkapjuk a kiválogatást tételét. $\mathbf{X}_i$-t akkor adjuk hozzá az **Y** tömbhöz, ha **T** tulajdonságú, különben nem adjuk hozzá („semmit adunk hozzá").

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

Kimenet: $Y \in H^N$

Előfeltétel: –

Utófeltétel: $\forall i (1 \le i \le N)$: $T(X_i) \to X_i \in Y$ és nem $T(X_i) \to X_i \notin Y$

```
Kiválogatás(N,X,Y):
  Y:=üres
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Y:=végére(Y,X[i]) különben {nincs teendő}
  Ciklus vége
Eljárás vége.
```

Megszámolásból kiindulás:

```
Megszámolás(N,X,Db):
  Db:=0
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Db:=Db+1
  Ciklus vége
Eljárás vége.
```

A két algoritmus összekötése:

```
Kiválogatás(N,X,Db,Y):
  Db:=0
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Db:=Db+1; Y[Db]:=X[i]
  Ciklus vége
Eljárás vége.
```

Vagy egy másik megközelítés:

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $T: H \to L$,
$f: H^* \times H \to H$,
$f(x,e) := \begin{cases} \text{végére}(x,e), & T(e) \\ x, & \text{egyébként} \end{cases}$

Kimenet: $Y \in H^N$

Előfeltétel: –

Utófeltétel: $\forall i (1 \le i \le N)$: $Y_i = f(X, X_i)$

```
Kiválogatás(N,X,Y):
  Y:=üres
  Ciklus i=1-től N-ig
    Y:=f(Y,X[i])
  Ciklus vége
Eljárás vége.
```

Az f függvény törzsének behelyettesítése után kapjuk a végleges változatot:

```
Kiválogatás(N,X,Y):
  Y:=üres
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Y:=végére(Y,X[i]) különben {nincs teendő}
  Ciklus vége
Eljárás vége.
```

Szétválogatás (két kiválogatás)

A kiválogatásnál megismert tömbre értelmezett feltételes összeadás műveletét két feltételre alkalmazzuk. Ha $\mathbf{X_i}$ **T** tulajdonságú, akkor hozzáadjuk az **Y** tömbhöz, különben a **Z** tömbhöz.

Bemenet: $N \in \mathbb{N}$, $X \in H^N$, $T:H \rightarrow \mathbb{L}$

Kimenet: $Db \in \mathbb{N}$, $Y, Z \in \mathbb{N}^N$

Előfeltétel: –

Utófeltétel: $Db = \sum_{i=1}^{N} 1_{T(X_i)}$ és

$\forall i(1 \le i \le Db)$: $T(X_i)$ és $\forall i(1 \le i \le N\text{-}Db)$: nem $T(X_i)$ és $Y \subseteq (1,2,\ldots,N)$ és $Z \subseteq (1,2,\ldots,N)$

```
Szétválogatás(N,X,Db,Y,Z):
  Db:=0
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Db:=Db+1; Y[Db]:=i
  Ciklus vége
  DbZ:=0
  Ciklus i=1-től N-ig
    Ha nem T(X[i]) akkor DbZ:=DbZ+1; Z[DbZ]:=i
  Ciklus vége
Eljárás vége.
```

A fenti algoritmusban az azonos lépésszámú, független ciklusok, valamint az azonos feltételű elágazások összevonhatók. Így kapjuk a szétválogatás jól ismert algoritmusát.

```
Szétválogatás(N,X,Db,Y,Z):
  Db:=0; DbZ:=0
  Ciklus i=1-től N-ig
    Ha T(X[i]) akkor Db:=Db+1; Y[Db]:=i
            különben DbZ:=DbZ+1; Z[DbZ]:=i
  Ciklus vége
Eljárás vége.
```

Tehát sorozatszámításból lett a másolás, a másolásból a kiválogatás, a kiválogatásból a szétválogatás, azaz a szétválogatás is a sorozatszámításból származik.

Metszet (kiválogatásban eldöntés)

A metszet tétel gyakorlatilag a kiválogatás és az eldöntés tétel összeépítése. Mindkettőről beláttuk már, hogy a sorozatszámítás tételéből levezethetőek.

Bemenet: $N,M \in \mathbb{N}$, $X \in H^N$, $Y \in H^M$

Kimenet: $Db \in \mathbb{N}$, $Z \in H^{\min(N,M)}$

Előfeltétel: halmaz(X,N) és halmaz(Y,M)

Utófeltétel: $Db = \sum_{\substack{i=1 \\ X_i \in Y}}^{N} 1$ és $\forall i(1 \le i \le Db)$: $Z_i \in X$ és $Z_i \in Y$ és halmaz(Z,Db)

Definíció: halmaz:$H^* \times \mathbb{N} \to \mathbb{L}$

halmaz(h,n)=$\forall i,j(1 \le i \ne j \le n)$: $i \ne j \to h_i \ne h_j$

```
Metszet(N,X,M,Y,Db,Z):
  Db:=0
  Ciklus i=1-től N-ig
    Ha eleme?(X[i],Y) akkor Db:=Db+1; Z[Db]:=X[i]
  Ciklus vége
Eljárás vége.
```

```
eleme?(x,Y):
  j:=1
  Ciklus amíg j≤M és x≠Y[j]
    j:=j+1
  Ciklus vége
  eleme?:=(j≤M)
Függvény vége.
```

Függvény beillesztésével

```
Metszet(N,X,M,Y,Db,Z):
  Db:=0
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg j≤M és X[i]≠Y[j]
      j:=j+1
    Ciklus vége
    Ha j≤M akkor Db:=Db+1; Z[Db]:=X[i]
  Ciklus vége
Eljárás vége.
```

Unió (másolás + kiválogatásban eldöntés)

Az unió tétel gyakorlatilag a másolás, a kiválogatás és az eldöntés tétel összeépítése. Mindháromról beláttuk már, hogy a sorozatszámítás tételéből levezethetőek.

Bemenet: $N,M \in \mathbb{N}$, $X \in H^N$, $Y \in H^M$

Kimenet: $Db \in \mathbb{N}$, $Z \in H^{N+M}$

Előfeltétel: halmaz(X,N) és halmaz(Y,M)

Utófeltétel: $Db = N + \sum_{\substack{j=1 \\ y_j \in X}}^{M} 1$ és $\forall i(1 \le i \le Db)$: $Z_i \in X$ vagy $Z_i \in Y$ és halmaz(Z,Db)

```
Unió(N,X,M,Y,Db,Z):
  Ciklus i=1-től N-ig
    Z[i]:=X[i]
  Ciklus vége
```

```
  Db:=N
  Ciklus j=1-től M-ig
    Ha nem eleme?(Y[j],X) akkor Db:=Db+1; Z[Db]:=Y[j]
  Ciklus vége
Eljárás vége.
```

Függvény beillesztésével, tömbök értékadásával

```
Unió(N,X,M,Y,Db,Z):
  Z:=X; Db:=N
  Ciklus j=1-től M-ig
    i:=1
    Ciklus amíg i≤N és X[i]≠Y[j]
      i:=i+1
    Ciklus vége
    Ha i>N akkor Db:=Db+1; Z[Db]:=Y[j]
  Ciklus vége
Eljárás vége.
```

Következtetések

A fentiekben beláttuk, hogy az összetett programozási tételek (másolás, kiválogatás, szétválogatás, metszet, unió) is egyetlen programozási tételre, a sorozatszámításra vezethetők vissza, amit pedig egy egyszerű összegzésből kaptunk.

Azért tudjuk a másolást, kiválogatás és szétválogatás tételét levezetni a sorozatszámításból, mert a másolás a tömbre értelmezett összeadást használja (azaz üres tömbbel indulunk, és ahogy feldolgozzuk X elemeit, úgy adjuk hozzá Y-hoz, gyakorlatilag ez az a „végére" művelet, amelyre a másolás tételénél hivatkoztunk). A kiválogatás annyiban különbözik ettől, hogy ott feltételesen használjuk a tömbre értelmezett összeadást. (A kapcsolat hasonló, mint az előző cikkünkben az összegzés és a feltételes összegzés között). Ha a kiválogatás levezethető, akkor levezethető a szétválogatás is, illetve, mivel a metszet és az unió is az előzőekre épül, ezért azok is levezethetőek a sorozatszámítás tételéből.

Az összetett tételek közül nem foglalkoztunk a rendezéssel. A rendezés különböző algoritmusai ugyanis más-más programozási tételekre vezethetők vissza, pl. a minimumkiválasztásos rendezés a minimumkiválasztásra és a (helyben) másolásra, a beillesztéses rendezés a keresésre és közben másolásra, a számlálva szétosztó rendezés a megszámolásra és a másolásra.

Nem jelenti mindez azonban azt, hogy felesleges e programozási tételek **önálló** megfogalmazása. Kezdő programozók ugyanis a programozási tételeket a feladattípusokból ismerik fel, amelyek az összetett tételek esetén megfelelnek ezeknek az alaptételeknek [1,5].

Ez a dolgozat pedig arról szól, hogy elvi szempontból elég belátnunk az első programozási tétel helyességét, mert abból az összes többit levezethetjük. Tehát, ha a sorozatszámítás helyes, akkor az összes többi is az.

Irodalom

[1] Péter SZLÁVI, László ZSAKÓ: *Módszeres programozás.* Műszaki Könyvkiadó, Budapest. 1986.

[2] Tibor GREGORICS: *Programozás – tervezés.* ELTE-Eötvös Kiadó, 2013.

[3] Ákos FÓTHI: *Bevezetés a programozáshoz*. Fóthi Ákos, 2012. (http://people.inf.elte.hu/fa/pdf/konyv.pdf − utolsó megtekintés 2017.05.05.)

[4] Tibor GREGORICS: *Programming theorems on enumerator*. Teaching Mathematics and Computer Science 8/1, 2010. (http://tmcs.math.unideb.hu/load_doc.php?p=186&t=abs − utolsó megtekintés 2017.05.05.)

[5] Péter SZLÁVI, László ZSAKÓ at al.: *Programozási alapismeretek*. Online tananyag, ELTE Informatikai Kar, 2012. (http://progalap.elte.hu/downloads/seged/eTananyag/ − utolsó megtekintés 2017.05.05.)

[6] Péter SZLÁVI, László ZSAKÓ at al.: *Módszeres programozás: programozási tételek*. Mikrológia 19. ELTE Informatikai Kar, 2008.

[7] Péter SZLÁVI, Gábor TÖRLEY, László ZSAKÓ: *Programming theorems have the same origin*. XXX. DIDMATTECH 2017, Trnava University, Faculty of Education, 2017 (http://real.mtak.hu/55421/1/SzP_TG_ZsL_Programming_Theorems_Have_the_Same_Origin.pdf − utolsó megtekintés 2017.05.05.)