9. előadás

Programozás-elmélet

Programozási feladatok megoldásakor a top-down (strukturált) programtervezés esetén három vezérlési szerkezetet használunk:

- szekvencia
- elágazás
- ciklus

Eddig megismertük az alábbi fogalmakat:

- állapottér ($A = A_1 \times \ldots \times A_n$)
- változó ($a_i \in A_i$)
- feladat ($F \subseteq A \times A$)
- program ($S \subseteq A \times A^{**}$)
- programfüggvény (a programfutás eredménye)

($p(S) \subseteq A \times A$)

Egy S program megoldja az F feladatot, ha $D_F \subseteq D_{p(S)}$ és $\forall a \in D_F : p(S)(a) \subseteq F(a)$. A "megoldás" ellenőrzésére és a programhelyesség részleges bizonyítására bevezettük az elő- és utófeltétel, valamint a leggyengébb előfeltétel fogalmát: Legyen $Q, R : A \to \mathbb{L}$ két állítás, S program. A $\{Q\}\, S\, \{R\}$ szimbólummal jelöltük azt, hogy $\forall a \in [Q]$ esetén $p(S)(a) \subseteq [R]$. A $[Q]$ előfeltétel halmazt felfoghatjuk az állapottér azon részeként, amelyre meg akarjuk a feladat megoldását kapni. Az elérendő eredmény megadására szolgál az $[R]$ utófeltétel halmaz. A $\{Q\}\, S\, \{R\}$ szimbólummal jelölt implikáció igazolása jelenti a parciális programhelyesség igazolását. Az $\mathit{lf}\,(S, R)$ leggyengébb előfeltétel a legbővebb Q előfeltételt jelenti.

Egy feladat megadása (specifikációja) a következőképpen történhet:

- bemenő (input) adatok
- kimenő (output) adatok
- a feladatban használt fogalmak definíciója
- az eredmény kiszámítási szabálya
- a bemenő adatokra kirótt feltételeket (előfeltételek)
- a kimenő adatokra kirótt feltételek (utófeltételek).

Lényeges észrevétel: a feladatok osztályokba sorolhatók a feladat jellege szerint és a feladatosztályokhoz a teljes feladatosztály megoldó algoritmusokat tudunk készíteni. Ezeket a tipus feladatokat *programozási tételeknek* nevezzük, mert igazolható, hogy megoldásaik a feladat garantáltan helyes megoldásai.
A programozási tételek ismeretében a "legtöbb" feladat megoldása során "csak" a megfelelő programozási tételt kell alkalmazni. Így elvileg helyes (de nem mindig optimális) megoldást kapunk.
A feladatok olyanok, hogy egy, vagy több adatsokasághoz rendelünk valamilyen eredményt. Egyszerűség kedvéért feltesszük, hogy ezek sorozatok, amelyeket tömbbel ábrázolhatunk. Vizsgált feladatosztály típusok:

- egy sorozathoz egy értéket rendelő feladatok
- egy sorozathoz egy sorozatot rendelő feladatok
- egy sorozathoz több sorozatot rendelő feladatok
- több sorozathoz egy sorozatot rendelő feladatok.

A feladatok általános alakja:

Input	:	$n \in \mathbb{N}_0,\ x \in H^n,\ F : H^* \to G$
Output	:	$S \in G$
Q	:	–
R	:	$S = F(x_1, \ldots, x_n)$

Egy bemenő sorozattól függő S értéket kell meghatároznunk. A feladatcsoporthoz több feladattípus tartozik.

Néhány példa:
F1. Adjuk meg n alkalmazottból álló osztály bértömegét a bérek ismeretében!
F2. Egy m elemű betűsorozat betűit füzzük össze egyetlen szövegtipusú változóba!
F3. A Balatonnál k madarász végez megfigyeléseket. Mindegyik megadta, hogy milyen madarakat látott. Készítsünk algoritmust, amely megadja a Balatonnál látott madarakat!
F4. Adjuk meg az első n természetes szám szorzatát!

A feladatokban közös: adatok sorozatához rendelünk egy értéket, amelyet, egy az egész sorozaton értelmezett függvény ad meg (n szám összege, m betű konkatenációja, k halmaz uniója, n szám szorzata).

Az algoritmus:

Változók:

n : egész

{a feldolgozandó sorozat elemszáma}

x : tömb(1..n:elemtipus)

{a feldolgozandó sorozat elemei}

F_0 : elemtipus$_1$ {a művelet nulleleme}

S : elemtipus$_2$ {az eredmény}

Input	:	$n \in \mathbb{N}_0$, $x \in H^n$, $F : H^* \to G$
Output	:	$S \in G$
Q	:	$\exists F_0 \in G$ (nullelem) és $\exists f : G \times H \to G$ és
		$F(x_1, \ldots, x_n) = f(F(x_1, \ldots, x_{n-1}), x_n)$ és $F() = F_0$
R	:	$S = F(x_1, \ldots, x_n)$

A feladat lehetséges variációi:

$$F = \sum, \prod, \cup, \cap, \vee, \wedge, \& \text{ (konkatenáció)}$$
$$F_0 = 0, 1, [], \text{alaphalmaz}, \textit{igaz}, \textit{hamis}, "".$$

A megoldás indukció segítségével:
- $i = 0$ esetre tudjuk az eredmény: F_0,
- ha $(i-1)$-re tudjuk az F_{i-1} eredményt, akkor az i-re az eredményt $F_i = f(F_{i-1}, x_i)$ adja $(i = 1, \ldots, n)$.

A feladatot megoldó algoritmus:

sorozatszámítás(n, x, s)
  $s := F_0$
  for $i = 1 : n$
    $s := f\,(s, x\,(i))$
  end
eljárás vége

Az F1. feladat esetén az eljárás:

sorozatszámítás(n, x, s)
  $s := 0$
  for $i = 1 : n$
    $s := s + x(i)$
  end
eljárás vége

Az F2. feladat esetén az eljárás:

sorozatszámítás(m, x, sz)
  $sz :=$ "" {üres szöveg}
  for $i = 1 : m$
    $sz := sz\&x(i)$
  end
eljárás vége

Az F3. feladat esetén:

unio(k, X, H)
  $H := []$
  for $i = 1 : k$
    $H := H \cup X\,(i)$
  end
eljárás vége

Végül az F4. feladat esetén:

sorozatszámítás(n, x, p)
  $p := 1$
  for $i = 1 : n$
    $p := p * x\,(i)$
  end
eljárás vége

A feladat annak eldöntése, hogy

a) van-e a sorozatban adott tulajdonságú elem

b) a sorozat mindegyik eleme rendelkezik-e ezzel a tulajdonsággal.

Az a) esetben a feladat alakja:

Input	:	$n \in \mathbb{N}_0,\ x \in H^n,\ T : H \to \mathbb{L}$
Output	:	$VAN \in \mathbb{L}$
Q	:	–
R	:	$VAN \equiv (\exists i\ \ (1 \leq i \leq n) : T(x_i))$

Az algoritmus:

Függvény:

T : *elemtípus* $\to \mathbb{L}$

Változók:

n : egész — a feldolgozandó sorozat elemszáma

x : tömb (1..n:elemtipus) — a feldolgozandó sorozat elemei

VAN : logikai — az eredmény

A feladattípus megoldására az előző programozási tétel is alkalmas az $F = \vee$, $f = \vee$, $F_0 = hamis$ megfeleltetéssel:

eldöntés(n, X, S)
  $S := hamis$
  for $i = 1 : n$
    $S := S \vee T(x(i))$
  end
eljárás vége

Vegyük észre, hogy ha egyszer S igaz lesz, akkor igaz is marad. Tehát nem kell a ciklust végig számolni. Eszerint az igazi megoldó algoritmus:

eldöntés(n, x, VAN)
  $i := 1$
  while $(i \leq n) \wedge (\neg T(x(i)))$
    $i := i + 1$
  end
  $VAN := (i \leq n)$
eljárás vége

A b) esetben a megoldást az adja, hogy átfogalmazzuk:

$$\forall i : 1 \leq i \leq n : T(x_i) \equiv \nexists i : 1 \leq i \leq n \wedge \neg T(x_i).$$

A sorozatszámítás itt az $F = \wedge$, $f = \wedge$, $F_0 = igaz$ értékekkel lehet alkalmazni.

A megoldó algoritmus:

eldöntés$(n, x, MIND)$
  $i := 1$
  while $(i \leq n) \wedge (T(x(i)))$
    $i := i + 1$
  end
  $MIND := (i > n)$
eljárás vége

Példa: Döntsük el, hogy egy adott sorozat monoton növekedő-e! Esetünkben $T(x_i) \equiv (x_i \leq x_{i+1})$ és ezért a ciklus csak $(n-1)$-ig mehet. A megoldás:

eldöntés$(n, x, MONOTON)$
  $i := 1$
  while $(i \leq n-1) \wedge (x(i) \leq x(i+1))$
    $i := i+1$
  end
  $MONOTON := (i > n-1)$
eljárás vége

A feladat sorozat adott tulajdonságú elemének (indexének) megadása, amelynek létezését feltesszük:

Input	:	$n \in \mathbb{N},\ x \in H^n,\ T : H \to \mathbb{L}$
Output	:	$SORSZ \in \mathbb{N}$
Q	:	$\exists i\ (1 \leq i \leq n) : T\,(x_i)$
R	:	$(1 \leq SORSZ \leq n) \wedge T\,(x_{SORSZ})$

Az algoritmus:

Függvény:

T : *elemtípus* $\to \mathbb{L}$

Változók:

n : egész {a feldolgozandó sorozat elemszáma}

x : tömb(1..n:elemtipus) {a feldolgozandó sorozat elemei}

$SORSZ$: egész {az eredmény}

A megoldás hasonlít az előző feladattípushoz:

kiválasztás$(n, x, SORSZ)$
  $i := 1$
  while $\neg T(x(i))$
    $i := i + 1$
  end
  $SORSZ := i$
eljárás vége

Feladat: A program az első T tulajdonságú elemet adja meg. Hogyan kell módosítani, hogy az utolsó ilyet adja meg?

A feladat adott tulajdonságú elem megadása egy sorozatban, ha van ilyen. Ha nincs, akkor ezt a tényt kell megadni. Tehát az előző két tétel mindegyikét tartalmazza:

Input	:	$n \in \mathbb{N},\ x \in H^n,\ T : H \to \mathbb{L}$
Output	:	$VAN \in \mathbb{L},\ SORSZ \in \mathbb{N}$
Q	:	–
R	:	$VAN \equiv (\exists i\ \ (1 \leq i \leq n) : T(x_i)) \wedge$
		$\wedge\, (VAN \Longrightarrow (1 \leq SORSZ \leq n) \wedge T(x_{SORSZ}))$

Az algoritmus:

Függvény:

$T : elemtípus \to \mathbb{L}$

Változók:

n : egész {a feldolgozandó sorozat elemszáma}

x : tömb(1..n:elemtipus) {a feldolgozandó sorozat elemei}

VAN : logikai {az eredmény - van-e megfelelő elem}

$SORSZ$: egész {az eredmény -a megfelelő elem sorszáma}

A megoldó algoritmust az eldöntés algoritmusa adja kiegészítve a kiválasztási algoritmus megfelelő részével:

keresés$(n, x, VAN, SORSZ)$
  $i := 1$
  while $(i \leq n) \wedge (\neg T(x(i)))$
    $i := i + 1$
  end
  $VAN := (i \leq n)$
  if VAN **then** $SORSZ := i$
eljárás vége

A feladat annak meghatározása, hogy hány adott tulajdonságú elem van egy sorozatban:

Input	:	$n \in \mathbb{N}_0$, $x \in H^n$, $T : H \to \mathbb{L}$, $\chi : H \to \{0,1\}$
		$\chi(x) = 1$, ha $T(x)$ és $\chi(x) = 0$, ha $\neg T(x)$
Output	:	$DB \in \mathbb{N}_0$
Q	:	–
R	:	$DB = \sum_{i=1}^{n} \chi(x_i)$

Az algoritmus:

Függvény:

T : *elemtípus* $\to \mathbb{L}$

Változók:

n : egész {a feldolgozandó sorozat elemszáma}

x : tömb(1..n:elemtipus) {a feldolgozandó sorozat elemei}

DB : egész {az eredmény -a megfelelő elemek száma}

A feladat egy sorozatszámítás (tkp. összegzés a χ függvény értékeire). A megoldó algoritmus:

megszámolás(n, x, DB)
  $DB := 0$
  for $i = 1 : n$
    if $T(x(i))$ **then** $DB := DB + 1$
  end
eljárás vége

A feladat az, hogy válasszuk ki egy sorozat legnagyobb (legkisebb) elemét:

Input	:	$n \in \mathbb{N}_0$, $x \in H^n$, H rendezett halmaz ($\exists <, \leq$ reláció)
Output	:	$MAX \in \mathbb{N}$
Q	:	$n \geq 1$
R	:	$1 \leq MAX \leq n \wedge \forall i \ (1 \leq i \leq n) : x_{MAX} \geq x_i$

Az algoritmus:

Változók:

n : egész {a feldolgozandó sorozat elemszáma}

x : tömb(1..n:elemtipus) {a feldolgozandó sorozat elemei}

MAX : egész {a maximális értékű elem sorszáma}

A sorozatszámításnál $F(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n)$ és

$$f(x, y) = \max(x, y)$$

függvényeket használjuk és az $F_0 = x_1$ választást (Nem neutrális elem, de az indexeléssel kompenzáljuk)! A megoldó algoritmus:

maximumkiválasztás(n, x, MAX)
  $MAX := 1$
  for $i = 2 : n$
    if $x(MAX) < x(i)$ **then** $MAX := i$
  end
eljárás vége

Ha a maximális értéket is akarjuk, akkor az algoritmus:

maximumkiválasztás$(n, x, MAX, MEXERT)$
  $MAX, MAXERT := 1, x(1)$
  for $i = 2 : n$
    if $MAXERT < x(i)$ **then** $MAX, MAXERT := i, x(i)$
  end
eljárás vége

A legkisebb érték meghatározásához a " $<$ " relációt át kell írni a " $>$ " relációra (A változóneveket is célszerű átírni a *MIN*, *MINERT* nevekre).