Egyszerű programok kezdőknek

V. rész

Az Euklideszi algoritmusról

Az *euklideszi algoritmus* egy számelméleti algoritmus, amely két szám legnagyobb közös osztóját határozza meg.

Két szám *legnagyobb közös osztója* a két szám azon közös osztója, amely minden közös osztónak többszöröse.

Az *a*, *b* számok legnagyobb közös osztójának szokásos jelölése a magyar szakirodalomban (*a*, *b*) vagy lnko(*a*, *b*); az angol irodalomban gcd(*a*, *b*). Például: lnko(12, 18) = 6, lnko(20, 15) = 5, lnko(–21, 3) = 3.

A legnagyobb közös osztó hagyományos megkereséséhez meg kell határozni az adott két szám törzstényezőit, azaz a számokat fel kell bontani prímszámok szorzatára (az egyes prímszámok bizonyos hatványon szerepelhetnek). Például az lnko(60, 24) = lnko($2^2 \cdot 3 \cdot 5$, $2^3 \cdot 3$). A törzstényezős felbontásból vesszük a legkisebb hatványon szereplő közös tagokat, ezek szorzata jelenti a legnagyobb közös osztót. A fenti példánál ez $2^2 \cdot 3$, vagyis 12.

A legnagyobb közös osztónak ez a meghatározási módszere csak kis számok esetén működik jól, nagy számoknál sok időt vesz igénybe a számítási folyamat (a törzstényezőre bontás).

Egy sokkal elegánsabb és hatékonyabb algoritmus az *euklideszi algoritmus*, amely nevét az ókori görög matematikusról, Eukleidészről (Kr. e. 300 körül született) kapta. Helyesen tehát *eukleidészi algoritmus* lenne a neve, de ebben a kifejezésben hagyományosan rögzült a név *euklideszi* alakban.

A viszonylag egyszerű algoritmusnak két alakja is van. Az első úgy működik, hogy első lépésében maradékosan osztjuk *a*-t *b*-vel, a második lépésben *b*-t a maradékkal, majd az előbbi maradékot az új maradékkal, és így tovább, tehát mindig az osztót osztjuk a maradékkal.

Formálisan leírva:

```
adott a, b
amíg b > 0 végezd el
    r := a mod b
    a := b
    b := r
(amíg) vége
eredmény a
```

C nyelvre pedig egyszerűen így írható át:

```
int main()
{
  int a, b, r;
  scanf("%i", &a);
  scanf("%i", &b);
  while(b>0)
  {
     r=a%b;
     a=b;
```

```
    b=r;
  }
  printf("%i\n", a);
}
```

Az algoritmus második alakjában a maradékszámítást ismétléses kivonással helyettesítjük, formálisan tehát:

adott *a, b*
ha *a = 0* ***akkor eredmény*** *b*
*(**ha**)* ***vége***
amíg *b > 0* ***végezd el***
    ha *a > b* ***akkor*** *a := a – b*
    különben *b := b –a*
        *(**ha**)* ***vége***
*(**amíg**)* ***vége***
eredmény *a*

C nyelvre így írható át az algoritmus:

```
int main()
{
  int a, b;
  scanf("%i", &a);
  scanf("%i", &b);
  if(a==0) printf("%i\n", b);
  while(b>0)
    if(a>b) a-=b;
    else b-=a;
  printf("%i\n", a);
}
```

Az algoritmus rekurzívan is leírható a következő egyszerű rekurzív képlet alapján:

$$\text{lnko}(a,b) = \begin{cases} a, \text{ ha } b = 0 \\ \text{lnko}(b, a \bmod b), \text{különben} \end{cases}$$

vagy a kivonásos változatban:

$$\text{lnko}(a,b) = \begin{cases} b, \text{ ha } a = 0 \\ \text{lnko}(a-b,\ b), \text{ ha } a > b \\ \text{lnko}(a,\ b-a), \text{ ha } b \geq a \end{cases}$$

Az első megvalósítása *C*-ben:

```
int lnko(int a, int b)
{
  if(b==0) return a;
  return lnko(b, a%b);
}
```

A legnagyobb közös osztó fogalma kiterjeszthető véges elemszámú számsorozatokra is: a *legnagyobb közös osztó* véges sok szám olyan közös osztója (azaz olyan szám, amely a véges sok szám mindegyikét osztja), amely bármely más közös osztónál nagyobb.

Tehát például: lnko(*a*, *b*, *c*) = lnko(*a*, lnko(*b*, *c*)).

Így véges sok számra a következő rekurzív függvényt írhatjuk meg (*oszd meg és uralkodj* / *divide et impera* elvre épülve):

```
int lnkos(int *a, int k, int v)
```

```
{
  if(v-k==0) return a[k];
  if(v-k==1) return lnko(a[k],a[v]);
  return   lnko(lnkos(a,   k,   (k+v)/2),lnkos(a,
(k+v)/2+1, v));
}
```

Megfigyelhetjük, hogy ha a sorozat egy elemű (v–k=0, ahol v a számsorozat végső, k pedig a számsorozat kezdő indexe), akkor a legnagyobb közös osztó maga a szám, ha kételemű, akkor meghatározzuk ennek a két számnak a legnagyobb közös osztóját, különben indítjuk a rekurzív számítási folyamatot.

A függvényt például így hívjuk meg:

```
int main()
{
  int a[6] = {4, 24, 8, 4,
16, 32};
  printf("%i\n", lnkos(a, 0,
5));
}
```

Kovács Lehel István

Táplálkozási kérdések vegyész szemmel

A glutamát kimutatására alkalmazott módszerek bioszenzorokkal

A felvett táplálékot az élő szervezet részben az elhasznált anyagainak pótlására, az élő anyaga gyarapítására (növekedés), más részét a különféle élettevékenységek energiaszükségletének fedezésére használja fel. A táplálkozás mértékét a szervezet szükséglete, illetve a felvett tápanyagok tápértéke szabja meg. A tápértéket az adott tápanyag energiaszolgáltatásának mértékével szokás jellemezni. Kaloriméterben való elégetésekor 1 grammnyi tápanyag által szolgáltatott energia kilokalória egységben kifejezve a következő: zsír 9,3; szénhidrát 4,1; fehérje 5,6. Az emberi szervezet számára nem közömbös hogy a felsorolt, fő tápanyagokból milyen arányban részesül. A tápláléknak mindig kell tartalmaznia megfelelő mennyiségű és minőségű fehérjét, mert ebből szerzi meg a szervezet azokat az aminosavakat, amelyeket önmaga nem képes előállítani (esszenciális aminosavak). Az egészséges felnőtt ember átlagos napi igénye a fő tápanyagokból: 70 g fehérje, 50 g zsír és 500 g szénhidrát. E mellett vízre, konyhasóra, nyomelemekre és vitaminokra van szüksége a szervezetnek, melyek hiánya különböző élettani zavarokat okozhat. Tehát mondhatjuk, hogy a táplálkozás az élet fenntartásához szükséges. De mi is az élet? A gondolkodók az emberiség története során próbálkoztak a meghatározásával, de a mai napig sem sikerült tökéletesen. A természettudományok mai fejlettségi szintjén a legelfogadhatóbb élet-definíciót Günter von Kiedrowski a következő módon fogalmazta meg (2002-ben): „az élő szervezetek olyan, működésükben összekapcsolt, helyi, nem lineáris, információsan ellenőrzött kémiai rendszerek populációját képezik, melyek képesek önreprodukcióra, alkalmazkodásra és együttes fejlődésre, amelynek révén a működési összetettség magasabb globális szintjeit érik el”. Ez az átfogó jellemzése