Ég és Föld vonzásában – a természet titkai

Informatikai tehetséggondozás:

Programozási tételek összeépítése

TÁMOP-4.2.3.-12/1/KONV

A projektek az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósulnak meg.

Gyakran előfordul, hogy programozási tételeket egymás után kell használnunk. Ezen egymásutániságnál azonban sok esetben a két megoldó algoritmus egybeépítése egyszerűbb, rövidebb, hatékonyabb megoldást eredményez. Ebben a részben ezekkel foglalkozunk. Az egymásra építés mindig két programozási tétel összefogását jelenti, a fejezeteket a korábban alkalmazandó tétel szerint fogalmaztuk meg.

1. Másolással összeépítés

Másolással bármelyik programozási tétel egybeépíthető, hiszen csupán annyi a teendő, hogy a programozási tételben szereplő X(i) bemenő adatra hivatkozást kicseréljük g(X(i))-re.

Ha például egy számsorozat elemeinek négyzetösszegét kellene megadnunk, az egy *másolást* (számokhoz számok négyzetei rendelése) és egy *összegzés*t tartalmaz. Nézzük meg e két tétel általános egymásra építését!

A megoldásban – mint azt az összegzésnél tettük – induljunk ki a nullelemből, alkalmazzuk a f függvényt az addig kiszámított értékre és a sorozat eleméből kiszámított értékre!

```
Másolás_összegzés(N,X,S):
  S:=0
  Ciklus I=1-től N-ig
    S:=S+g(X(I)))
  Ciklus vége
Eljárás vége.
```

Második példaként vegyük a *másolás* és a *maximumkiválasztás* összeépítését! Ebben a maximális elem értékét és az indexét is meghatározzuk. (Az előző feladat analógiájára ilyen lehet a legnagyobb abszolút értékű szám abszolút értékének meghatározása egy számsorozatból.)

Ebben a megoldásban – a maximumkiválasztás alapján – vegyük az első elemből kiszámított függvényértéket induló maximumnak, ezt hasonlítsuk a további elemekből kiszámított függvényértékekkel, és a legnagyobbat őrizzük meg!

```
Másolás_maximumkiválasztás(N,X,MAX,MAXERT):
  MAX:=1; MAXERT:=g(X(1))
  Ciklus I=2-től N-ig
    Ha MAXERT<g(X(I)) akkor MAXERT:=g(X(I)); MAX:=I
  Ciklus vége
Eljárás vége.
```

2. Megszámolással összeépítés

A *megszámolást* három elemi programozási tétellel érdemes egybeépíteni, az *eldöntéssel,* a *kiválasztással,* valamint a *kereséssel*.

Itt olyan kérdéseket tehetünk fel, hogy van-e egy sorozatban legalább K db T tulajdonságú elem, adjuk meg a sorozat K-adik T tulajdonságú elemét stb.

Az általánossága miatt nézzük a *megszámolás* és a *keresés* egymásra építését! Az *eldöntésnél,* illetve a *kiválasztásnál* hasonlóan kellene eljárnunk, hiszen e két típusalgoritmus megoldásszövege része a *keresés* megoldásszövegének.

Induljunk ki a keresés megoldásából! A keresés ciklusa akkor állt le, amikor megtaláltuk az első T tulajdonságú elemet. Ezt kell kicserélni arra, hogy csak a K.-nál álljon le (ha egyáltalán van K). A ciklusmagban viszont számolnunk kell a T tulajdonságú elemeket – ahogyan azt a *megszámolás*nál tettük!

```
Megszámolás_Keresés(N,X,K,VAN,SORSZ):
  I:=0; DB:=0
  Ciklus amíg I<N és DB<K
    I:=I+1
    Ha T(X(I)) akkor DB:=DB+1
  Ciklus vége
  VAN:=(DB=K)
  Ha VAN akkor SORSZ:=I
Eljárás vége.
```

3. Maximumkiválasztással összeépítés

*Maximumkiválasztás*sal kapcsolatban azt a kérdést fogalmazhatjuk meg, hogy hány darab maximális értékű elem van, s hogy melyek a maximális értékű elemek.

Itt tehát a maximumkiválasztást kell egybeépíteni a megszámolással, illetve a kiválogatással.

Az a kérdés, hogy van-e egyáltalán több maximális értékű elem, egy menetben nem dönthető el, azaz a három tételt nem lehet egymásba építeni, hanem csak egymás után alkalmazni.

A korábbiakban megállapítottuk, hogy a kigyűjtéses *kiválogatás* mindig tartalmaz egy *megszámolást,* így csak a *kiválogatással* kell foglalkoznunk.

Az összeépítés alapgondolata, hogy a lokális maximumok megőrzésével együtt válogassuk is ki a lokális maximumokat. Természetesen új lokális maximum megtalálásakor a korábbi kigyűjtést el kell felejteni. A kiválasztó ciklus végén a lokális maximum éppen a keresett maximumérték, tehát az éppen kigyűjtött sorszámok a maximumok sorszámai lesznek.

```
Maximumkiválogatás(N,X,DB,Y,MAXERT):
  MAXERT:=X(1); DB:=1; Y(DB):=1
  Ciklus I=2-től N-ig
    Elágazás
      X(I)>MAXERT  esetén MAXERT:=X(I); DB:=1; Y(DB):=I
      X(I)=MAXERT  esetén DB:=DB+1; Y(DB):=I
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

Feladatatok programozási tételekre a Nemes Tihamér OITV-ről és az Informatika OKTV-ről

1. feladat

Egy lövészversenyen a versenyzők egymás után lőnek. Ismerjük N (1≤N≤1000) versenyző eredményét. Készíts programot (`lovesz.pas`, …), amely beolvassa N értékét és az N db eredményt, majd megadja:

A. minden versenyzőre, hogy az addig szereplők közül hányan értek el nála jobb eredményt;

B. azokat a versenyzőket, akik a verseny valamelyik időszakában álltak az első helyen;

C. azokat a versenyzőket, akik a verseny valamelyik időszakában álltak az utolsó helyen;

D. a verseny győzteseit!

Példa:

Bemenet:	Kimenet:
N=6	Jobb eredmény: 0 0 2 0 3 3
1. versenyző: 594	Állt az első helyen: 1 2 4
2. versenyző: 596	Állt az utolsó helyen: 1 3
3. versenyző: 582	Győztesek: 4
4. versenyző: 599	
5. versenyző: 590	
6. versenyző: 590	

A. NEM ÖSSZEÉPÍTÉS

B. az összes maximum számításakor egy új maximális érték megtalálása esetén ne töröljük a korábbi maximumokat;

C. az összes minimum számításakor egy új maximális érték megtalálása esetén ne töröljük a korábbi minimumokat;

D. az összes maximum számításakor egy új maximális érték megtalálása esetén töröljük a korábbi maximumokat!

```
B(bdb,bt):
  max:=1; bdb:=1; bt(1):=1
  Ciklus i=2-től n-ig
    Ha pont(i)≥pont(max) akkor bdb:=bdb+1; bt(bdb):=i
    Ha pont(i)>pont(max) akkor max:=i
  Ciklus vége
Eljárás vége.
```

```
C(cdb,ct):
  max:=1; cdb:=1; ct(1):=1
  Ciklus i=2-től n-ig
    Ha pont(i)≤pont(max) akkor cdb:=cdb+1; ct(cdb):=i
    Ha pont(i)<pont(max) akkor max:=i
  Ciklus vége
Eljárás vége.
```

```
D(ddb,dt):
  max:=1; ddb:=1; dt(1):=1
  Ciklus i=2-től n-ig
    Ha pont(i)=pont(max) akkor max:=i; ddb:=1; dt(1):=i
    különben ha pont(i)>pont(max) akkor ddb:=ddb+1; dt(ddb):=i
  Ciklus vége
Eljárás vége.
```

2. feladat

G(x)-ek összege

Egy kártyajátékban az egyes lapoknak számértékük van. Minden lapot egy színnel és egy figurával adunk meg. A színek: piros, zöld, tök, makk. A figurák: 7-es, 8-as, 9-es, 10-es, alsó, felső, király, ász. A számot tartalmazó figurák annyit érnek, amennyi a ráírt szám. Az alsó 2-t, a felső 3-at, a király 4-et, az ász 11-et ér. A piros lapoknál az értéket duplán kell számítani.

Készíts programot (`kartya.pas`,…), amely beolvassa egy játékos N ($1 \leq N \leq 4$) kártyáját, majd megadja, hogy a lapok összesen hány pontot érnek!

Példa

Bemenet:	Kimenet:
Kártyák száma? 3 1. kártya színe? piros 1. kártya figurája? alsó 2. kártya színe? tök 2. kártya figurája? 7-es 3. kártya színe? tök 3. kártya figurája? ász	A kártyák értéke: 22

Tároljuk a `szin` vektorban a lehetséges kártyaszíneket, a pirossal kezdve; a `figura` vektorban pedig a kártyafigurákat úgy, hogy az értékük éppen az indexük legyen (üresen hagyva azokat az indexű elemeket, amilyen értékű figura nincs)!

A feladat megoldása a lapok értékeinek összegzése, ahol az értéket egy speciális függvénnyel számítjuk ki:

```
Kártya:
  e:=0
  Ciklus i=1-től n-ig
    j:=1
    Ciklus amíg s(i)≠szin(j)
      j:=j+1
    Ciklus vége
    Ha j=1 akkor sz:=2 különben sz:=1
    j:=1
    Ciklus amíg f(i)≠figura(j)
      j:=j+1
    Ciklus vége
    e:=e+sz*j
  Ciklus vége
Eljárás vége.
```

3. feladat

Van N darab egységnyi méretű négyzetlapunk. KxK-as négyzeteket kell összerakni belőlük, először a lehető legnagyobbat, utána a maradékból egyre kisebbeket,...

Írj programot (NEGYZETEK.PAS, NEGYZETEK.C, …), amely beolvassa a négyzetlapok számát (1≤N≤10000), majd megadja, hogy a fenti elven mekkora négyzetek rakhatók ki belőlük!

Példa:

```
  N=72  ⇒  8*8-as négyzet, 2*2-es négyzet, 2*2-es négyzet
```

A feladat megoldása speciális függvénnyel számolt értékek összegzése.

Meg kell találni az N előtti legnagyobb négyzetszámot, majd ezt levonva N-ből újra kezdeni ezt az eljárást. Ezt mindaddig végezzük, amíg N-re 0-t nem kapunk.

```
Négyzet(N):
  k:=1
  Ciklus amíg (k+1)*(k+1)≤N
    k:=k+1
  Ciklus vége
```

```
  Ciklus amíg N>0
    Ki: k; N:=N-k*k
    Ciklus amíg k*k>N
      k:=k-1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

4. feladat

Ismerjük az A (1≤A≤32767) pozitív egész számot, valamint azt is tudjuk, hogy az A számjegyeinek összege valamilyen számrendszerben éppen S. A számrendszer alapszáma biztosan nem nagyobb A+1-nél.

Készíts programot (*SZAMREN.PAS*, *SZAMREN.C*, …), amely beolvassa az adatokat, majd kiírja a képernyőre, hogy az A szám számjegyeinek összege mely számrendszerekben lehet éppen S!

Példa:

```
    Bemenet:  A=127, S=3
    Kimenet: Lehetséges számrendszer=5, 63, 125
```

Magyarázat: 127=**1***5³+**0***5²+**0***5+**2**, 127=**2***63+**1**, 127=**1***125+**2**

A megoldás egy speciális elemekből álló összeg kiszámolása, ahol az elemeket magukat is számoljuk. Egy szám számjegyei összegét A alapú számrendszerben a következőképpen számoljuk ki:

```
összeg(i,A):
  b:=0
  Ciklus amíg a>0
    b:=b+a mod i; a:=a div i
  Ciklus vége
  összeg:=b
Függvény vége.
```

Ezután a feladat megoldása:

```
Számrendszer(S):
  Ciklus i=2-től A+1-ig
    Ha összeg(i,A)=S akkor Ki: i
  Ciklus vége
Eljárás vége.
```

5. feladat

Egy nemzetközi vonat több napon keresztül megy az egyik végállomásáról a másik végállomására. Ismerjük minden állomásra az érkezési időt. A vonat a végállomás kivételével minden állomáson pontosan 10 percet várakozik, majd továbbindul.

Készíts programot (*ALLOMAS.PAS* vagy *ALLOMAS.C*), amely beolvassa az állomások számát (2≤N≤100), a kezdő állomásról indulási időt (0≤óra≤23, 0≤perc≤59), majd pedig a további N-1 állomásra érkezési időt (óra, perc). A program ezekből számítsa ki, hogy

A. hány perc volt a leghosszabb időszak, amikor a vonat sehol sem állt meg;

B. a vonat mely állomások között haladt (vagy mely állomásokon állt) éjfélkor!

Példa:

```
  Bemenet: N=7; indulás=9 óra 20 perc; érkezések: (13,30),
(19,45), (4,00), (16,30), (23,55), (6,30).
  Leghosszabb menetidő: 740 perc {a 4. és az 5. állomás között}
```

A. speciális értékek maximumát kel meghatározni (a menetidőbe a várakozási időt nem szabad beszámítani);

B. NEM TÉTEL ÖSSZEÉPÍTÉS.

Érdemes az indulási és érkezési időket percre átszámítani (`idő(i)`). Ekkor a szomszédos értékek különbsége (figyelembe véve a 10 perces várakozást) maximuma az első részfeladat megoldása.

A második részfeladatban azok a vonatok állnak éjfélkor valamelyik állomáson, amelyek éjfél előtt legfeljebb 10 perccel érkeztek, s azok haladnak két állomás között, amelyek az egyik állomáson még éjfél előtt voltak, a másikra pedig éjfél után érkeztek.

```
VonatokA(N,idő,legh,dba,a,dbh,h):
  legh:=idő(2)-idő(1)
  Ciklus i=3-tól N-ig
    Ha idő(i)>idő(i-1)      {ugyanazon a napon vannak-e?}
       akkor Ha idő(i)-idő(i-1)-10>legh
                akkor legh:=idő(i)-idő(i-1)-10
    különben Ha idő(i)+24*60-idő(i-1)-10>legh
                akkor legh:=idő(i)+24*60-idő(i-1)-10
  Ciklus vége
Eljárás vége.
```

6. feladat

Egy hegymászó a tervezett útvonala mentén méterenként megmérte a felszín tengerszint feletti magasságát. N helyen kapott mérési adatokat. Emelkedőnek nevezzük azt a számsorozatot, amelynek minden eleme nagyobb, mint az előtte levő. Az emelkedő helye az ilyen számsorozat első és utolsó tagjának sorszáma, a hossza pedig a számsorozatban levő számok darabszáma. (Emelkedő lehet balról jobbra, illetve jobbról balra haladva is!)

Készíts programot (HEGY.PAS vagy HEGY.BAS vagy HEGY.C), amely megadja, hogy az út során hol volt a leghosszabb emelkedő! (Ha több egyforma van, közülük egyet kell megadni.) Ha nincs emelkedő az út során, akkor írja ki, hogy NINCS!

A feladat megoldásához először be kell olvasni a mérések számát (1≤N≤100), majd pedig az N darab mérést; majd ezután ki kell írni az eredményt.

Példa:

```
Bemenet:N=10
        Mérések: 100,110,115,110,105,115,125,130,125,125
Kimenet: 5,8
```

A feladat megoldásához meg kell határozni az oda- és a visszaúton is az emelkedők kezdetét és végét, s közben közülük ki kell választani a leghosszabbat. Ha nincs emelkedő, akkor a leghosszabb emelkedő kezdete 0 marad.

```
Leghosszabb emelkedő(n,h):
  maxk:=0; maxv:=0; k:=0
  Ciklus i=2-től n-ig
    Ha k=0 és h(i)>h(i-1) akkor k:=i-1
    Ha k>0 és h(i)≤h(i-1) akkor
       Ha i-k>maxv-maxk+1 akkor maxk:=k; maxv:=i-1
       k:=0
    Elágazás vége
  Ciklus vége
  Ha k>0 akkor Ha n+1-k>maxv-maxk+1 akkor maxk:=k; maxv:=n
  k:=0
  Ciklus i=n-1-től 1-ig
    Ha k=0 és h(i)>h(i+1) akkor k:=i+1
    Ha k>0 és h(i)≤h(i+1) akkor
      Ha k-i>|maxk-maxv|+1 akkor maxk:=k; maxv:=i+1
      k:=0
    Elágazás vége
  Ciklus vége
  Ha k>0 akkor Ha k>|maxk-maxv|+1 akkor maxk:=k; maxv:=1
  Nincs:=(maxk=0)
Eljárás vége.
```

7. feladat

A Villamos-közlekedési Vállalat (VKV) felmérést végzett a villamosok kihasználásáról, melyet számítógéppel kell feldolgozni. A villamos-vonalon N állomás van, beleértve az induló- és a végállomást is. Egy út során a villamosvezetőnek meg kellett számolnia minden állomáson a fel- és a leszállókat, s neked ezekből az adatokból kell adott jellemzőket kiszámolnod.

Készíts programot (VILLAMOS.PAS, vagy VILLAMOS.C vagy VILLAMOS. BAS), amely beolvassa N ($2 \le N \le 100$) értékét, a vezető által adott számokat (N*2 adat, mindegyik pozitív), majd belőlük a következőket határozza meg és írja ki a képernyőre:

A. Hány ember utazott összesen a villamoson?

B. Mely állomásokon szállt le a villamosról az összes utas?

C. Mi volt a villamoson a maximális utas szám?

D. Hány állomásközi szakaszt tett meg a villamos úgy, hogy egyetlen utas sem volt rajta?

<u>Példa:</u>

```
Bemenet: 5 állomás
  Felszállók: 5 3 0 2 0
  Leszállók:  0 4 4 0 2
A: Összesen 10 ember utazott a villamoson.
B: Mindenki leszállt a 3. és az 5. állomáson.
C: A maximális utas szám 5 volt.
D: 1 szakaszon nem volt utas.
```

Jelölje `fel(i)` az i-edik állomáson felszállók, `le(i)` pedig a leszállók számát! Az egyik részfeladat a tartalmi helyesség ellenőrzése.

A. NEM TÉTEL ÖSSZEÉPÍTÉS – egyszerűen a felszállók számnak összege;

B. Legyen $\mathrm{utasszam}(i) = \sum_{j=1}^{i-1} \mathrm{fel}(j) - \sum_{j=1}^{i} \mathrm{le}(j)$! Ekkor a feladat a `utasszam (i)=0` értékek megtalálása;

C. A maximális `utasszam(i)+fel(i)` meghatározása a feladat

D. A `utasszam(i)+fel(i)=0` értékek száma a feladat.

Másodikként ki kell válogatni azon állomások sorszámát, amikor a leszállás után (a felszállás előtt) az utasszám 0 lett (`db`,`hely`). Ehhez természetesen számolni kell a pillanatnyi utasszámot (`utasszam`). Ez utóbbi maximuma lesz a C részfeladat megoldása (`max`). A D részfeladat megoldásához azon esetek számát kell megadni, amikor a felszállások után 0 volt az aktuális utasszám.

```
Villamos:
  utasszam:=0; max:=0; db:=0; uresek:=0
  Ciklus i=1-től n-ig
    utasszam:=utasszam-le(i)
    Ha utasszam=0 és i>1 és le(i)≠0
                    akkor db:=db+1; hely(db):=i
    utasszam:=utasszam+fel(i)
    Ha utasszam>max akkor max:=utasszam
    Ha utasszam=0 és i<n akkor uresek:=uresek+1
  Ciklus vége
Eljárás vége.
```

8. feladat

Ádám és Éva N játékot játszott egymással. Minden játékról tudjuk, hogy melyikük nyerte meg.

Készíts programot (`jatek.pas`, `jatek.c`, …), amely megadja, hogy Ádám vagy Éva nyert-e többször! (Páros N esetén elképzelhető, hogy egyik sem).

Példa:

```
Bemenet:                Kimenet:
  5                         Ádám – hamis
  Ádám                      Éva - igaz
  Éva
  Éva
  Éva
  Ádám
```

A feladat átfogalmazva: nyert-e Éva legalább N/2-ször, vagy nyert-e Ádám legalább N/2-ször. Ebből következően nem kell feltétlenül az összes eredményt végignéznünk.

```
Ádám_Éva(A,E):
  A:=hamis; E:=hamis;i:=0; DB:=0
  Ciklus amíg i<N és DB≤N/2 és i-DB≤N/2
    i:=i+1
    Ha X(i)='Ádám' akkor DB:=DB+1
  Ciklus vége
  A:=DB>N/2; E:=(i-DB)>N/2
Eljárás vége.
```