Érdekes informatika feladatok

XLIV. rész

Seherezádé dátumai

2020. február 2-ika, vagyis 2020/02/02 (éééé/hh/nn), egy olyan dátum, amely (az elválasztójeleket figyelmen kívül hagyva) balról jobbra vagy visszafelé, jobbról balra olvasva is ugyanaz.

Az ilyen szókapcsolatokat, mondatokat vagy számokat *palindrom*nak nevezzük.

A palindromszámokat Buckminster Fuller a *Szinergetika* című könyvében *Seherezádé-számok*nak hívta az 1001 éjszaka meséi miatt.

2020. február 2. tehát *Seherezádé-dátum*.

Ennek kapcsán ismerkedjünk meg jobban a perzsa naptárral!

A perzsa naptár 622-ben kezdődik, a perzsa év első napja (01/01) március 21. (szökőévben március 20.), *Noruz* (újév) napja maga a tavaszi napéjegyenlőség.

Az első hat hónap 31 napos, a következő öt 30 napos. Az utolsó hónap 29 napos, szökőévben pedig 30.

Szökőév minden néggyel osztható év, kivéve a százzal is oszthatókat. Szökőévek viszont a 400-zal osztható évek. John Herschel javaslata alapján, a pontosság kedvéért, minden 4000. év kivételesen nem szökőév.

a) Állítsuk elő a legkisebb és a legnagyobb Seherezádé-dátumot! A perzsa naptár alapján mondjuk meg, hogy hány nap különbség van köztük!
b) Adott két dátum éééé/hh/nn formátumban. Vizsgáljuk meg, hogy Seherezádé-dátumok-e, s ha igen, a perzsa naptár alapján mondjuk meg, hogy hány nap különbség van köztük!
c) Hány Seherezádé dátum van a legkisebb és a legnagyobb Seherezádé-dátum között?
d) Mekkora a legkisebb és a legnagyobb távolság két egymást követő Seherezádé-dátum között a perzsa naptár alapján (hány nap)? Ha több is van, akkor az elsőt kell kiírni.

Próbáljuk meg először a fenti kérdéseket a logika és matematika útján megválaszolni, nyilván némelyik esetében egyszerűbb programmal.

a) Legkisebb: 1001/10/01, legnagyobb: 9290/09/29. Köztük a perzsa naptár szerint 3 027 492 nap különbség van.
b) *Lásd program*. Például 2011/11/02 és 2020/02/02 között 3012 nap van.
c) 330 Seherezádé-dátum van összesen. Induljunk ki a dátumok végéből, vagyis a napokból, ezek 01, 02, …, 29, 31 lehetnek. Nem lehetnek 10, 20, 30, tehát 0-ra végződő napjaink, mert a palindrom miatt az az év kezdete lenne, és perzsa naptár esetleg csak 06-tal kezdődhet. Tehát 28 lehetőségünk van. 31-ikével csak 6 hónap végződhet az értelmezés szerint, a többivel pedig 12 hónap, így a Seherezádé-dátumok száma: $27 \times 12 + 6 = 330$. A szökőévnek itt nincs jelentősége, mert 30-ra nem végződhet Seherezádé-dátum.

d) Legkisebb különbség: 671 nap, 1010/01/01 és 1011/11/01 között, legnagyobb különbség: 259 698 nap, 2290/09/22 és 3001/10/03 között. A legnagyobb különbség meghatározásánál nyilván az évezred váltásokat kell figyelembe venni, itt a lehetőség a legnagyobb különbségre, tehát 1360/06/31 és 2001/10/02, 2290/09/22 és 3001/10/03, 3290/09/23 és 4001/10/04, 4290/09/24 és 5001/10/05 stb. dátumok közötti különbségeket kell vizsgálni, mivel minden évezredben a legnagyobb Seherezádé-dátum és a következő évezred legkisebb Seherezádé-dátuma egymásutániak és közöttük lehet a legnagyobb különbség. Az első esetben a különbség 234 213, a második esetben 259 698. Megfigyelhetjük, hogy ezután mindig ugyanennyi lesz a különbség, kivéve 4000 és 8000 esetében, mert azok nem szökőévek. Ebben a két esetben 259 697 a különbség. Mivel az elsőt kell kiíratni, így marad 2290/09/22 és 3001/10/03, a köztük lévő 259 698 nap különbséggel. A legkisebb különbség akkor fordulhat elő, ha minél kevesebb év van két egymást követő Seherezádé-dátum között (egy évben csak egy Seherezádé-dátum lehetséges). Ilyen esetek a következők: 1010/01/01 és 1011/11/01, 2010/01/02 és 2011/11/02, 3010/01/03 és 3011/11/03 stb. Ezen párok között pontosan 671 nap különbség van a perzsa naptár szerint, így mivel az elsőt kell megadni, a megoldás: 1010/01/01 és 1011/11/01.

Most pedig nézzük meg a feladat kapcsán megírt programot. Érdekes a dátumok kezelése, a perzsa naptár hónapjainak, napjainak a megadása.

Modulhasználat:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <sstream>

using namespace std;
```

Struktúra a dátumok kezelésére:

```
typedef struct
{
  unsigned int ev;
  unsigned short ho;
  unsigned short nap;
} Datum;
```

Dátumátalakító függvények (egészből dátumot, dátumból szöveget, szövegből dátumot):

```
Datum ToDatum(unsigned int ev, unsigned short ho, unsigned short
nap)
{
  Datum d;
  d.ev = ev;
  d.ho = ho;
  d.nap = nap;
  return d;
}
```

```
string DatumToString(Datum d)
{
  stringstream r;
  r << d.ev << "/";
  if(d.ho>9) r << d.ho;
  else r << "0" << d.ho;
  r << "/";
  if(d.nap>9) r << d.nap;
  else r << "0" << d.nap;
  string s = r.str();
  if(s.length() < 10) s = "0" + s;
  return s;
}

Datum StringToDatum(string d)
{
  Datum r;
  r.ev = atoi(d.substr(0, 4).c_str());
  r.ho = atoi(d.substr(5, 2).c_str());
  r.nap = atoi(d.substr(8, 2).c_str());
  return r;
}
```

Képlet a szökőévekre:

```
bool szokoev(unsigned int ev)
{
  return ((ev%4==0)&&(ev%100!=0))||((ev%400==0)&&(ev%4000!=0));
}
```

Napok száma egy évben:

```
int EvNap(unsigned int ev)
{
  return 365 + (szokoev(ev)?1:0);
}
```

Hány napos egy hónap a perzsa naptár szerint? Bemenet: az év és a hónap. Kimenet a hónap napjainak száma. Ha ezt a függvényt átírjuk a saját naptárunk szerint, akkor a program arra is működni fog:

```
unsigned short PerzsaHoNap(unsigned int ev, unsigned short ho)
{
  if(ev<622) return 0;
  if(ho<1||ho>12) return 0;
  switch (ho)
  {
    case 1: return 31;
    case 2: return 31;
    case 3: return 31;
    case 4: return 31;
    case 5: return 31;
    case 6: return 31;
    case 7: return 30;
    case 8: return 30;
    case 9: return 30;
    case 10: return 30;
    case 11: return 30;
```

```
      case 12: return szokoev(ev)?30:29;
    }
  }
```

És akkor álljon itt a saját naptárunkra:

```
  unsigned short HoNap(unsigned int ev, unsigned short ho)
  {
    if(ho<1||ho>12) return 0;
    switch (ho)
    {
      case 1: return 31;
      case 2: return szokoev(ev)?29:28;
      case 3: return 31;
      case 4: return 30;
      case 5: return 31;
      case 6: return 30;
      case 7: return 31;
      case 8: return 31;
      case 9: return 30;
      case 10: return 31;
      case 11: return 30;
      case 12: return 31;
    }
  }
```

Helyes-e egy szövegként megadott perzsa dátum?

```
  bool jodatum(string a)
  {
    Datum d;
    if(a.length()!=10) return false;
    if(a[0]<'0'||a[0]>'9') return false;
    if(a[1]<'0'||a[1]>'9') return false;
    if(a[2]<'0'||a[2]>'9') return false;
    if(a[3]<'0'||a[3]>'9') return false;
    if(a[4]!='/') return false;
    if(a[5]<'0'||a[5]>'1') return false;
    if(a[6]<'0'||a[6]>'9') return false;
    if(a[7]!='/') return false;
    if(a[8]<'0'||a[8]>'3') return false;
    if(a[9]<'1'||a[9]>'9') return false;
    d.ev = atoi(a.substr(0, 4).c_str());
    if(d.ev<622||d.ev>9999) return false;
    d.ho = atoi(a.substr(5, 2).c_str());
    if(d.ho<1||d.ho>12) return false;
    d.nap = atoi(a.substr(8, 2).c_str());
    if(d.nap<1) return false;
    if(d.nap>PerzsaHoNap(d.ev, d.ho)) return false;
    return true;
  }
```

Ellenőrzi, hogy egy szövegként megadott dátum Seherezádé-dátum-e vagy sem:

```
  bool seherezade(string a)
  {
    return                                                    (jo-
datum(a)&&a[0]==a[9]&&a[1]==a[8]&&a[2]==a[6]&&a[3]==a[5]);
  }
```

Egy nagy számból szöveges dátumot állít elő, nagyon hasznos a generálásoknál, ciklusban tudunk végigmenni a dátumokon:

```
string makedatum(unsigned long long szam)
{
  string d = "0000/00/00";
  if(szam<6220101||szam>99999999) return d;
  stringstream mystream;
  mystream << szam;
  d = mystream.str();
  if(d.length()<7) d = "0" + d;
  d = d.insert(4, "/");
  d = d.insert(7, "/");
  return d;
}
```

A következőkben két dátum közötti különbség (napok számában mérve) kiszámításához szükséges, hogy iteratív és rekurzív függvényeket írjunk meg. Az első függvény visszatéríti, hogy két teljes év között hány nap telt el:

```
int EvOsszeg(unsigned int ev1, unsigned int ev2)
{
  int o = 0;
  for(unsigned int i = ev1; i <= ev2; ++i)
    o += EvNap(i);
  return o;
}
```

A következő függvény eredménye, hogy ugyanazon év két teljes hónapja között hány nap telt el a perzsa naptár szerint:

```
int HoOsszeg(unsigned int ev, unsigned short ho1, unsigned short ho2)
{
  int o = 0;
  for(unsigned short i = ho1; i <= ho2; ++i)
    o += PerzsaHoNap(ev, i);
  return o;
}
```

Rekurzív függvény, amely visszatéríti, hogy két dátum szerint hány nap telt el a perzsa naptár szerint:

```
int sz(Datum dmin, Datum dmax)
{
  if(dmax.ev==dmin.ev&&dmax.ho==dmin.ho&&dmax.nap==dmin.nap)    return
  0;
  if(dmax.ev==dmin.ev&&dmax.ho==dmin.ho) return dmax.nap - dmin.nap;
  if(dmin.ev == dmax.ev)
    return  sz(dmin,  ToDatum(dmin.ev,  dmin.ho,  PerzsaHoNap(dmin.ev,
    dmin.ho))) +
           1 + HoOsszeg(dmin.ev, dmin.ho + 1, dmax.ho - 1) +
           sz(ToDatum(dmax.ev, dmax.ho, 1), dmax);
  return sz(dmin, ToDatum(dmin.ev, 12, PerzsaHoNap(dmin.ev, 12))) +
         1 + EvOsszeg(dmin.ev + 1, dmax.ev - 1) +
         sz(ToDatum(dmax.ev, 1, 1), dmax);
}
```

A fenti függvény a következőképpen működik:

- Ha a két dátum megegyezik, akkor közöttük 0 nap telt el.
- Hogy ha a két dátum esetén csak a napok különböznek (ugyanaz az év, ugyanaz a hónap), akkor a két dátum között a különbség a napok közötti különbség.
- Ha a két dátum esetén az évek megegyeznek, akkor a különbséget az eltelt hónapok és napok adják, vagyis rekurzívan megvizsgáljuk, hogy hány nap telt el a kisebb dátum és a kisebb dátum hónapjának a végéig, hónapot váltunk (+ 1 nap), majd a nagyobbik dátum hónapját megelőző hónapig összeadjuk a napok számát (ezek a teljes hónapok), ezután pedig rekurzívan megvizsgáljuk, hogy hány nap telt el a nagyobbik dátum hónapjából. Ez az összeg adja meg az eltelt napok számát. Pont úgy számolunk, mintha egy naptár segítségével, kézzel számolnánk ki, hogy hány nap telt el ugyanabban az évben két dátum között.
- Ha a két dátum esetén az évek is különböznek, akkor a következőképpen járunk el:
  - Rekurzívan megvizsgáljuk, hogy hány nap van a kisebb év végéig.
  - Évet váltunk (+ 1 nap).
  - A következő évtől a nagyobbik évet megelőző évig összeadjuk az években lévő napok számát.
  - Rekurzívan megvizsgáljuk, hogy a nagyobbik évben hány nap telt el január elsejétől a megadott második dátumig.
  - A fentieket összeadva megkapjuk a két dátum között eltelt napok számát.

A következő függvény két tetszőleges sorrendben megadott dátum közötti különbséget téríti vissza (nem számít, hogy melyik év van előbb):

```
int kulonbseg(Datum d1, Datum d2)
{
  Datum dmin, dmax;
  int nap = 0;
  if(d1.ev < d2.ev) {dmin = d1; dmax = d2;}
  else if(d1.ev > d2.ev) {dmin = d2; dmax = d1;}
  else if(d1.ho < d2.ho) {dmin = d1; dmax = d2;}
  else if(d1.ho > d2.ho) {dmin = d2; dmax = d1;}
  else if(d1.nap < d2.nap) {dmin = d1; dmax = d2;}
  else if(d1.nap > d2.nap) {dmin = d2; dmax = d1;}
  else {dmin = d1; dmax = d1; return 0;}
  return sz(dmin, dmax);
}
```

A fenti függvényeket felhasználva most már megírhatjuk azt a főprogramot, amely választ ad az a), b), c), d) kérdéseinkre:

```
int main()
{
  //a), c) - az a) és c) kérdést egyszerre kezelhetjük
  ofstream ki("evek.txt");
  int c = 0;
  for(long long i = 6220101; i <= 99999999; ++i)
    if(seherezade(makedatum(i)))
    {
      ki<<makedatum(i)<<endl;
      ++c;
    }
```

```
       ki.close();
       cout<<c<<" datum van."<<endl;
       cout<<kulonbseg(StringToDatum("1001/10/01"),          StringTo-
Datum("9290/09/29"))<<endl;

       //b)
       cout<<kulonbseg(StringToDatum("2011/11/02"),          StringTo-
Datum("2020/02/02"))<<endl;

       //d)
       int min, max;
       Datum d1, d2;
       Datum dmin1, dmin2, dmax1, dmax2;
       string s1, s2;
       ifstream be("evek.txt");
       be>>s1;
       be>>s2;
       d1 = StringToDatum(s1);
       d2 = StringToDatum(s2);
       min = kulonbseg(d1, d2);
       max = min;
       dmin1 = d1;
       dmin2 = d2;
       dmax1 = d1;
       dmax2 = d2;
       while(!be.eof())
       {
         be>>s1;
         d1 = StringToDatum(s1);
         if(d1.ev==d2.ev&&d1.ho==d2.ho&&d1.nap==d2.nap) continue;
         if(kulonbseg(d1, d2) < min)
         {
            min = kulonbseg(d1, d2);
            dmin1 = d1;
            dmin2 = d2;
         }
          if(kulonbseg(d1, d2) > max)
         {
            max = kulonbseg(d1, d2);
            dmax1 = d1;
            dmax2 = d2;
         }
         s2 = s1;
         d2 = d1;
       }
       be.close();
       cout<<"Legkisebb    kulonbseg:    "<<min<<"    nap,    "<<Datum-
ToString(dmin1)<<", "<<
           DatumToString(dmin2)<<" kozott."<<endl;
       cout<<"Legnagyobb    kulonbseg:    "<<max<<"    nap,    "<<Datum-
ToString(dmax1)<<", "<<
           DatumToString(dmax2)<<" kozott."<<endl;
       return 0;
     }
```

Kovács Lehel István