Cuprins

Emanuela Cerchez, Marinel Şerban

PROGRAMAREA ÎN LIMBAJUL C/C++ PENTRU LICEU

●

Ediția a II-a revăzută și adăugită

POLIROM
2021

4.11. *Aplicații*

Temperaturi

Se citesc de la tastatură temperaturile înregistrate la Institutul meteorologic în fiecare dintre zilele lunii ianuarie. Scrieți un program care să numere în câte zile ale lunii au fost temperaturi strict pozitive și de câte ori s-a schimbat semnul temperaturilor (s-a înregistrat o trecere de la temperaturi pozitive la temperaturi negative sau invers) în decursul lunii ianuarie.

Soluție

Vom memora temperaturile din luna ianuarie într-un vector `T` cu `31` de elemente. Vom parcurge vectorul de la primul până la ultimul element, testând pentru fiecare element din vector dacă este o temperatură pozitivă. În caz afirmativ, numărăm încă o zi cu temperaturi pozitive (în acest scop utilizăm variabila `nrp`).

Pentru a determina numărul de schimbări de semn din vectorul `T` este suficient ca în timpul parcurgerii să verificăm dacă există în vector două elemente situate pe poziții consecutive și de semne contrare. Schimbările de semn vor fi numărate în variabila `nrs`.

```
#include <iostream>
#define NMax 31
using namespace std;
int main()
{int T[NMax], nrp, nrs=0, i;
 for (i=0; i<31; i++) cin>>T[i];
 nrp=T[0]>0?1:0;
 for (i=1; i<31; i++)
     {if (T[i]>0) nrp++;
      if (T[i-1]*T[i]<0) nrs++;     }
 cout<<"nr de zile cu temperaturi pozitive "<<nrp<<'\n';
 cout<<"nr de schimbari de semn "<<nrs<<'\n';
 return 0; }
```

Inserare medii

Se citește de la tastatură un număr natural `n` (`1<n≤50`) și elementele unui vector cu `n` componente reale. Să se insereze între oricare două elemente consecutive ale vectorului media lor aritmetică.

Soluție

O primă idee este de a citi elementele vectorului și apoi de a insera între oricare două elemente vecine media lor. Pentru a insera media, trebuie să-i „facem loc", adică trebuie să deplasăm toate elementele care urmează cu o poziție la dreapta. Nu uitați că, prin inserare, numărul de elemente din vector crește!

```
#include <iostream>
using namespace std;
int main ()
{float a[100];
 int n, i, j;
 cin >> n;
 for (i=0; i<n; i++) cin>>a[i];
 for (i=1; i<n; i+=2) //inserez media dintre a[i] si a[i-1]
   {//mut elementele de la i la n-1 cu o pozitie la dreapta
    for (j=n; j>i; j--) a[j]=a[j-1];
    a[i]=(a[i]+a[i-1])/2;        //plasez media pe pozitia i
    n++; }              //maresc numarul de elemente din vector
 for (i=0; i<n; i++) cout <<a[i]<<' ';
 return 0;}
```

O soluție mai simplă ar fi fost de a plasa în vector elementele citite numai pe poziții de indice par, astfel încât pozițiile de indice impar să rămână disponibile pentru medii, evitând astfel deplasările succesive:

```
#include <iostream>
using namespace std;
int main ()
{float a[100];
 int n, i;
 cout <<"n="; cin >> n;
 for (i=0; i<n; i++) cin>>a[2*i];
 for (i=1; i<n; i++) a[2*i-1]=(a[2*i]+a[2*i-2])/2;
 for (i=0; i<2*n-1; i++) cout <<a[i]<<' ';
 return 0; }
```

Ciurul lui Eratostene

Să se genereze toate numerele prime mai mici decât o valoare specificată VMAX.

Soluție

O primă idee ar fi să parcurgem toate numerele naturale din intervalul [2, VMAX), pentru fiecare număr să verificăm dacă este prim și să afișăm numerele prime determinate.

O idee mai eficientă provine din Antichitate și poartă numele de ciurul lui Eratostene[23]. Ideea este de a „pune în ciur" toate numerele mai mici decât VMAX, apoi de a „cerne" aceste numere până rămân în ciur numai numerele prime. Mai întâi „cernem" (eliminăm din ciur) toți multiplii lui 2, apoi cernem multiplii lui 3 etc.

---

[23] Eratostene (276-196 î.e.n.) a fost un important matematician și filosof al Antichității. Deși puține dintre lucrările lui s-au păstrat, a rămas celebru prin metoda rapidă de determinare a numerelor prime și prin faptul că a fost primul care a estimat cu acuratețe diametrul Pământului.

Vom reprezenta ciurul ca un vector cu VMAX de componente care pot fi 0 sau 1, cu semnificaţia ciur[i]=0, dacă numărul i este în ciur, respectiv 1, în cazul în care i a fost eliminat din ciur.

Vom parcurge vectorul ciur de la 2 (cel mai mic număr prim), până la $\sqrt{VMAX}$. Dacă ciur[i] este 0, deducem că i este prim, dar toţi multiplii lui nu vor fi (deci eliminăm din ciur toţi multiplii lui i). Primul multiplu pe care îl eliminăm va fi i*i, deoarece toţi multiplii precedenţi au fost deja eliminaţi.

În final, în vectorul ciur vor avea valoarea 0 doar componentele de pe poziţii numere prime.

```
#include <iostream>
#define VMAX 1000000
using namespace std;
bool ciur[VMAX];
int main()
{int i, j;
 ciur[0]=ciur[1]=1; //0 si 1 nu sunt numere prime
 for (i=2; i*i<VMAX; i++)
     if (ciur[i]==0)     //i este prim
        //elimin toti multiplii lui i
        for (j=i*i; j<VMAX; j+=i)
            ciur[j]=1;
 for (i=2; i<VMAX; i++)
     if (ciur[i]==0) cout<<i<<' ';
return 0; }
```

Observaţie

Valoarea VMAX trebuie să fie suficient de mică astfel încât să fie posibilă declararea unui vector cu VMAX elemente de tip bool. În caz contrar, putem implementa ciurul lui Eratostene „pe biţi", întrucât pentru a memora 0 sau 1 nu este necesar un *byte*, este suficient un singur bit.

Nrdiv

Se consideră o secvenţă de N numere naturale nenule. Determinaţi numărul de divizori pentru fiecare număr din secvenţa dată.

Fişierul de intrare nrdiv.in conţine pe prima linie numărul natural N, care reprezintă numărul de valori din secvenţă. Pe următoarele N linii se află cele N numere naturale din secvenţă, câte un număr pe o linie.

Fişierul de ieşire nrdiv.out va conţine N linii. Pe linia i va fi scris numărul de divizori ai celui de-al i-lea număr din secvenţă (în ordinea din fişierul de intrare).

Restricții

- $1 \leq N \leq 50$
- $1 \leq$ numerele din secvență $\leq 10^{13}$

Exemple

nrdiv.in	nrdiv.out	**Explicații**
3 13 1 24	2 1 8	13 are doi divizori (1 și 13) 1 are un divizor (1) 24 are 8 divizori (1, 2, 3, 4, 6, 8, 12, 24)

Olimpiada Municipală de Informatică Iași, 2014

Soluție

O primă soluție ar fi ca pentru fiecare număr x să căutăm divizorii proprii de la 2 la sqrt(x), contorizând la fiecare divizor d găsit atât divizorul d, cât și complementarul său x/d, acordând o atenție specială pătratelor perfecte (numere care se divid prin radicalul lor și care au un număr impar de divizori). În concurs, această soluție ar obține un punctaj parțial din cauza depășirii limitei de timp.

O a doua soluție ar fi să descompunem fiecare număr x în factori primi. Dacă descompunerea în factori primi a lui x este $d_1^{p1}d_2^{p2}...d_k^{pk}$, atunci numărul de divizori ai lui x este $(p_1+1)(p_2+1)...(p_k+1)$.

Această soluție poate obține punctaj maxim doar dacă optimizăm descompunerea în factori primi. Cu ciurul lui Eratostene generăm toate numerele prime mai mici sau egale cu sqrt(VMAX) (unde VMAX este valoarea maximă din secvență). Reținem numerele prime într-un vector și la descompunerea în factori primi parcurgem vectorul de numere prime.

```
#include <fstream>
#include <cmath>
#define NMAX 52
#define VMAX 10000001
#define NRMAXPRIM 10000
using namespace std;
ifstream fin("nrdiv.in");
ofstream fout("nrdiv.out");
int N, nrprim;
long long int v[NMAX], vmax, prim[NRMAXPRIM];
bool ciur[VMAX];
int main()
{int i, p;
 long long int nr, d, j;
 fin>>N;
 for (i=0; i<N; i++)              //citire si determinare maxim
     {fin>>v[i];
      if (v[i]>vmax) vmax=v[i];    }
```

```
vmax=sqrt((double)vmax);      //ciur
for (d=2; d*d<=vmax; d++)
     if (!ciur[d])
          for (j=d*d; j<=vmax; j+=d) ciur[j]=1;
//transfer intr-un vector numerele prime <=vmax
prim[0]=2; nrprim=1;
for (d=3; d<=vmax; d+=2)
     if (!ciur[d]) prim[nrprim++]=d;
//descompunere in factori primi eficienta
for (i=0; i<N; i++)
    {nr=1;
     for (j=0; j<nrprim && prim[j]*prim[j]<=v[i]; j++)
          {
           for (p=0; v[i]%prim[j]==0; p++,v[i]/=prim[j]);
           nr*=(p+1);
          }
     if (v[i]>1) nr*=2;
     fout<<nr<<'\n';
    }
fout.close();
return 0; }
```

Copii

La ora de educaţie fizică, profesorul a cerut elevilor să se alinieze. Fiind un profesor pasionat de informatică, a observat imediat că există situaţii în care un copil de înălţime maximă este aşezat lângă un copil de înălţime minimă. Scrieţi un program care să determine câte perechi de elevi alăturaţi îndeplinesc această condiţie.

Soluţie

Vom reţine înălţimile copiilor, în ordinea în care s-au aliniat aceştia, în vectorul h. Determinăm mai întâi înălţimea celui mai înalt copil şi înălţimea celui mai scund copil. Parcurgem apoi vectorul, identificând toate perechile de copii alăturaţi în care primul copil are înălţimea maximă şi al doilea, înălţimea minimă sau invers.

```
#include <iostream>
#define NMax 40
using namespace std;
int main()
{int h[NMax], n, i, nr, min, max;
 cout<<"n= "; cin>>n;
 for (i=0; i<n; i++) cin>>h[i];
 //determinam inaltimea maxima si inaltimea minima
 for (max=min=h[0], i=1; i<n; i++)
     if (max<h[i]) max=h[i];
        else
        if (min>h[i]) min=h[i];
 //determinam numarul de perechi
```

```
for (nr=0, i=1; i<n; i++)
    if (h[i]==max&&h[i-1]==min ||h[i]==min&&h[i-1]==max)
        nr++;
cout<<nr<<'\n';
return 0; }
```

Exercițiu

Modificați programul precedent astfel încât să realimeze elevii pentru a obține un număr maxim de perechi de elevi alăturați cu proprietatea din enunț.

Perechi

Construiți un algoritm eficient care să determine toate perechile de numere naturale `a`, `b`, cu `a≤b`, având proprietatea că nu au nici o cifră comună și suma lor este egală cu `S`. Valoarea `S` este un număr natural citit de la tastatură (`S<100000000`). Fiecare pereche se va scrie pe un rând al ecranului, cu un spațiu între elementele ce compun perechea.

De exemplu, pentru `S=14`, se vor afișa (nu neapărat în această ordine) perechile:

```
3 11
6 8
4 10
5 9
0 14
```

Bacalaureat, august 2001

Soluție

O primă idee este de a genera toate perechile de numere mai mici sau egale cu `S` și pentru fiecare pereche să verificăm proprietatea cerută. Algoritmul de mai sus execută `s*(s-1)/2` operații, operațiile fiind relativ complexe. Valoarea maximă a variabilei `s` fiind foarte mare, timpul de execuție a acestui program va fi mare.

Algoritmul precedent poate fi transformat cu ușurință într-un algoritm liniar (care execută `s/2` operații), dacă observăm că valoarea lui `b` este unic determinată de valoarea lui `a` (`b=s-a`).

Pentru a testa dacă numerele `a` și `b` au cifre distincte, vom determina mulțimea cifrelor numărului `a`, apoi vom determina mulțimea cifrelor numărului `b`, apoi vom verifica dacă cele două mulțimi au elemente comune.

Pentru a determina mulțimea cifrelor numărului `a` vom considera un vector `cifrea` cu `10` componente (câte una pentru fiecare cifră din baza `10`), având semnificația `cifrea[i]=1`, dacă cifra `i` apare în numărul `a`, și `0`, în caz contrar. Inițial, toate componentele vectorului au valoarea `0`. Vom extrage succesiv cifrele numărului `a`, reținând pentru fiecare cifră extrasă valoarea `1` pe poziția corespunzătoare cifrei din vectorul `cifrea`. În mod analog procedăm și pentru numărul `b`, reținând mulțimea cifrelor în vectorul `cifreb`. După ce am construit cele două mulțimi de cifre, le