

# Cuprins

<b>1. Elemente de bază ale limbajului C/C++</b> .....	<b>13</b>
1.1. <i>Noțiuni introductive</i> .....	13
<i>Evoluția limbajelor de programare</i> .....	13
1.2. <i>Setul de caractere</i> .....	14
1.3. <i>Identificatori</i> .....	14
1.4. <i>Cuvinte rezervate</i> .....	15
1.5. <i>Comentarii</i> .....	15
1.6. <i>Separatori</i> .....	16
1.7. <i>Structura generală a unui program C/C++</i> .....	17
1.8. <i>Tipuri de date standard</i> .....	18
<i>Tipuri de date naturale</i> .....	18
<i>Tipuri de date întregi</i> .....	18
<i>Tipuri de date caracter</i> .....	20
<i>Tipuri reale</i> .....	21
<i>Tipul bool</i> .....	21
<i>Tipul void</i> .....	21
1.9. <i>Variabile</i> .....	22
1.10. <i>Preprocesare</i> .....	23
1.11. <i>Utilizarea funcțiilor din bibliotecile standard</i> .....	25
1.12. <i>Citirea/scrierea datelor</i> .....	26
1.13. <i>Citiri și scrieri în limbajul C++</i> .....	26
1.14. <i>Citiri și scrieri în limbajul C</i> .....	29
<i>Citirea datelor cu format specificat</i> .....	29
<i>Citirea caracterelor</i> .....	31
<i>Afișarea datelor cu format</i> .....	32
1.15. <i>Expresii</i> .....	34
<i>Operatori aritmetici</i> .....	34
<i>Operatori de incrementare/decrementare</i> .....	35

<i>Operatori relaționali</i> .....	35
<i>Operatori de egalitate</i> .....	36
<i>Operatori logici globali</i> .....	36
<i>Operatori logici pe biți</i> .....	37
<i>Operatori de atribuire</i> .....	38
<i>Operatori condiționali</i> .....	39
<i>Operatorul de referențiere (adresă)</i> .....	39
<i>Operatorul de conversie explicită</i> .....	40
<i>Operatorul de determinare a dimensiunii</i> .....	40
<i>Operatorul virgulă</i> .....	40
<i>Evaluarea expresiilor</i> .....	41
<i>Tabелul priorității operatorilor</i> .....	42
<b>1.16. Aplicații</b> .....	43
<i>Disc</i> .....	43
<i>Compus chimic</i> .....	43
<i>Triunghi</i> .....	44
<i>Schimb</i> .....	45
<i>Leul și iepurașul</i> .....	46
<i>Bazin</i> .....	47
<b>1.17. Probleme propuse</b> .....	48
<b>2. Instrucțiunile limbajului C/C++</b> .....	52
<b>2.1. Instrucțiunea expresie</b> .....	52
<b>2.2. Instrucțiunea compusă</b> .....	52
<b>2.3. Instrucțiunea if</b> .....	53
<b>2.4. Instrucțiunea switch</b> .....	53
<b>2.5. Instrucțiunea break</b> .....	54
<b>2.6. Instrucțiunea while</b> .....	55
<b>2.7. Instrucțiunea do-while</b> .....	56
<b>2.8. Instrucțiunea for</b> .....	57
<b>2.9. Aplicații</b> .....	58
<i>Modul</i> .....	58
<i>Paritate</i> .....	59
<i>Ecuatia de gradul al II-lea</i> .....	59
<i>Sumă</i> .....	60
<i>Aria triunghiului</i> .....	60
<i>Secvență</i> .....	61
<i>Calcul</i> .....	64
<i>Numărare</i> .....	66
<i>Putere</i> .....	68

<i>Numere</i> .....	69
<i>Coduri și caractere</i> .....	70
<i>Secvență simetrică</i> .....	71
<i>Maxim</i> .....	72
<i>Media aritmetică</i> .....	74
<i>Suma cifrelor</i> .....	74
<i>Perechi</i> .....	75
<i>Divizori</i> .....	76
<i>Număr prim</i> .....	77
<i>Descompunere în factori primi</i> .....	78
<i>Cel mai mare divizor comun</i> .....	80
<i>Gard</i> .....	81
<i>Exponent</i> .....	82
<i>Pitag</i> .....	83
<i>Canguri</i> .....	84
<i>Termen Fibonacci</i> .....	85
<i>Existență</i> .....	86
<i>Număr divizori primi</i> .....	87
<i>Expresie</i> .....	88
<i>Radical</i> .....	89
<i>Sumă de două numere prime</i> .....	90
<i>Progres</i> .....	91
<i>Cuvinte</i> .....	91
<i>Generare șir</i> .....	92
<i>Semne</i> .....	93
<i>Excursie</i> .....	94
<b>2.10. Probleme propuse</b> .....	95
<b>3. Fișiere</b> .....	108
<b>3.1. Noțiuni introductive</b> .....	108
<b>3.2. Fișiere text în limbajul C++</b> .....	109
<i>Declararea fișierelor</i> .....	109
<i>Deschiderea fișierelor</i> .....	109
<i>Citirea datelor dintr-un fișier</i> .....	111
<i>Scrierea datelor într-un fișier</i> .....	111
<i>Operații de test</i> .....	112
<i>Închiderea unui fișier</i> .....	113
<b>3.3. Fișiere text în limbajul C</b> .....	113
<i>Declararea fișierelor</i> .....	113
<i>Deschiderea fișierelor</i> .....	114
<i>Citirea datelor dintr-un fișier</i> .....	115
<i>Scrierea datelor într-un fișier</i> .....	115

Închiderea fișterelor.....	115
Aplicație.....	115
3.4. Probleme propuse .....	117
4. Tipuri structurate de date .....	118
4.1. Tablouri.....	118
4.2. Prelucrări elementare pe vectori.....	120
Citirea unui vector.....	120
Afișarea unui vector.....	120
Copierea unui vector.....	120
Determinarea elementului maxim/minim dintr-un vector.....	120
Media aritmetică a elementelor strict pozitive.....	121
Inversarea ordinii elementelor din vector.....	121
Verificarea unei proprietăți.....	122
Căutarea unui element într-un vector.....	122
Căutarea binară pe rezultat.....	125
Sortare.....	126
Interclasare.....	128
4.3. Prelucrări elementare cu matrice.....	129
Citirea unei matrice.....	129
Afișarea unei matrice.....	129
Parcurgerea unei matrice pe linii.....	129
Parcurgerea unei matrice pe coloane.....	130
4.4. Prelucrări elementare specifice matricelor pătratice.....	130
Parcurgerea diagonalelor.....	130
Parcurgerea elementelor de sub diagonala principală.....	131
Parcurgerea unei matrice pătratice pe chenare.....	131
4.5. Șiruri de caractere.....	132
4.6. Citirea și afișarea șirurilor de caractere.....	133
Citirea unui șir de caractere în limbajul C++.....	133
Citirea unui șir de caractere în limbajul C.....	134
Afișarea unui șir de caractere în limbajul C++.....	135
Afișarea unui șir de caractere în limbajul C.....	136
4.7. Tipul pointer.....	136
Declararea unui pointer de date.....	136
Operații cu pointeri.....	137
4.8. Legătura dintre pointeri și tablouri.....	139
Utilizarea funcțiilor standard de lucru cu șiruri de caractere.....	140
4.9. Tipul de date struct.....	146
Declararea unui tip struct.....	146

<i>Referirea la un câmp al unei structuri</i> .....	147
<i>Inițializarea unei structuri</i> .....	147
<i>Citirea și afișarea structurilor</i> .....	148
<i>Operații cu structuri</i> .....	148
<b>4.10. Asocierea unui nume pentru un tip de date</b> .....	148
<b>4.11. Aplicații</b> .....	149
<i>Temperaturi</i> .....	149
<i>Inserare medii</i> .....	149
<i>Ciurul lui Eratostene</i> .....	150
<i>Nrdiv</i> .....	151
<i>Copii</i> .....	153
<i>Perechi</i> .....	154
<i>Repetiție</i> .....	155
<i>Eliminare</i> .....	156
<i>Permutare ciclică</i> .....	158
<i>Generarea mulțimii</i> .....	159
<i>Subsecvență de sumă maximă</i> .....	161
<i>Panglica</i> .....	162
<i>Paint</i> .....	164
<i>Roua</i> .....	166
<i>Câștig</i> .....	169
<i>br</i> .....	171
<i>Uscat</i> .....	173
<i>Cărți</i> .....	175
<i>Marcare</i> .....	176
<i>Depozit</i> .....	177
<i>Cadrane</i> .....	178
<i>Matrice</i> .....	179
<i>Situație școlară</i> .....	180
<i>Pseudodiagonale</i> .....	183
<i>Submulțimi cu sume egale</i> .....	186
<i>Submatrix</i> .....	188
<i>Zid</i> .....	190
<i>Secvențe</i> .....	193
<i>Expresie</i> .....	195
<i>Furnici</i> .....	196
<i>Apariții cifră</i> .....	199
<i>Aparițiile unui cuvânt</i> .....	200
<i>Propoziție</i> .....	201
<i>Băcan</i> .....	202
<b>4.12. Probleme propuse</b> .....	204

<b>5. Metoda de programare Greedy</b> .....	214
<b>5.1. Prezentare generală</b> .....	214
<b>5.2. Aplicații</b> .....	214
<i>Problema spectacolelor</i> .....	214
<i>Problema rucsacului</i> .....	216
<i>Culmi</i> .....	217
<i>Reactivi</i> .....	218
<i>Venus</i> .....	220
<b>5.3. Probleme propuse</b> .....	223
<b>6. Algoritmi de tip succesori</b> .....	228
<b>6.1. Aplicații</b> .....	228
<i>Generare de submulțimi</i> .....	228
<i>Generare elemente produs cartezian</i> .....	229
<i>Generare permutări</i> .....	230
<i>Generare combinări</i> .....	231
<b>6.2. Probleme propuse</b> .....	232
<b>7. Stiva și coada</b> .....	236
<b>7.1. Stiva</b> .....	236
<i>Care este utilitatea stivelor?</i> .....	237
<i>Cum implementăm o stivă?</i> .....	237
<b>7.2. Aplicații cu stiva</b> .....	239
<i>Depou</i> .....	239
<i>Swap</i> .....	240
<i>Reacții</i> .....	242
<i>Manna-Pnueli</i> .....	244
<b>7.3. Algoritmii de Fill</b> .....	246
<i>Tsunami</i> .....	246
<b>7.4. Coada</b> .....	249
<i>Care este utilitatea unei cozi?</i> .....	250
<i>Cum implementăm o coadă?</i> .....	250
<i>Supermarket</i> .....	252
<b>7.5. Algoritmii lui Lee</b> .....	255
<i>Șoricel în labirint</i> .....	256
<b>7.6. Probleme propuse</b> .....	258
<b>8. Funcții</b> .....	265
<b>8.1. Subprograme în limbajul C/C++</b> .....	266
<b>8.2. Definiția unei funcții</b> .....	266

8.3. <i>Declararea funcțiilor</i> .....	268
<i>Utilitatea declarațiilor</i> .....	269
<i>Biblioteci de funcții și fișiere antet</i> .....	269
8.4. <i>Apelul funcțiilor</i> .....	270
8.5. <i>Transferul parametrilor prin referință</i> .....	272
<i>Utilizarea pointerilor ca parametri</i> .....	272
<i>Tipul referință</i> .....	273
8.6. <i>Variabile globale și variabile locale</i> .....	277
<i>Variabilele globale</i> .....	278
<i>Variabilele locale</i> .....	278
<i>Regula de omonimie</i> .....	278
<i>Operatorul de rezoluție</i> .....	278
8.7. <i>Specificatori de clasă de memorare</i> .....	279
8.8. <i>Parametrii funcției main ()</i> .....	280
8.9. <i>Caracteristici specifice funcțiilor C++</i> .....	282
<i>Funcții inline</i> .....	282
<i>Funcții cu parametri implicați</i> .....	282
8.10. <i>Proiecte</i> .....	283
8.11. <i>Aplicații</i> .....	284
<i>Cel mai mare divizor comun cu descompunere în factori primi</i> .....	284
<i>Replace</i> .....	286
<i>Incluziune</i> .....	288
<i>Operații cu numere naturale mari</i> .....	289
<i>Fotografie</i> .....	293
<i>Secvență regulată</i> .....	295
8.12. <i>Probleme propuse</i> .....	298
<b>Anexe</b> .....	307
1. <i>Tabela codurilor ASCII</i> .....	307
2. <i>Sisteme de numerație</i> .....	309
<i>Operații de conversie</i> .....	309
3. <i>Organizarea logică a memoriei</i> .....	311
4. <i>Reprezentarea datelor în memorie</i> .....	312
5. <i>Etapile dezvoltării programelor</i> .....	313
<b>Soluții și indicații</b> .....	317
<b>Bibliografie</b> .....	335

**Emanuela Cerchez, Marinel Șerban**

# **PROGRAMAREA ÎN LIMBAJUL**

# **C/C++**

## **PENTRU LICEU**



Ediția a II-a revăzută și adăugită

POLIROM  
2021



## 4.11. Aplicații

### Temperaturi

Se citește 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  $n$  ( $1 < n \leq 50$ ) 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  $V_{MAX}$ .

#### *Soluție*

O primă idee ar fi să parcurgem toate numerele naturale din intervalul  $[2, V_{MAX})$ , 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<sup>23</sup>. Ideea este de a „pune în ciur” toate numerele mai mici decât  $V_{MAX}$ , 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.

<sup>23</sup> 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  $V_{MAX}$  de componente care pot fi 0 sau 1, cu semnificația  $ciur[i]=0$ , 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  $\sqrt{V_{MAX}}$ . Dacă  $ciur[i]$  este 0, deducem că  $i$  este prim, dar toți multiplii lui nu vor fi (deci eliminăm din ciur toți multiplii lui  $i$ ). Primul multiplu pe care îl eliminăm va fi  $i^2$ , deoarece toți multiplii precedenți au fost deja eliminați.

În final, în vectorul  $ciur$  vor avea valoarea 0 doar componentele de pe poziții 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; }
```

### Observație

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

### Nrdiv

Se consideră o secvență de  $N$  numere naturale nenule. Determinați numărul de divizori pentru fiecare număr din secvența dată.

Fișierul de intrare `nrdiv.in` conține pe prima linie numărul natural  $N$ , care reprezintă numărul de valori din secvență. Pe următoarele  $N$  linii se află cele  $N$  numere naturale din secvență, câte un număr pe o linie.

Fișierul de ieșire `nrdiv.out` va conține  $N$  linii. Pe linia  $i$  va fi scris numărul de divizori ai celui de-al  $i$ -lea număr din secvență (în ordinea din fișierul de intrare).

*Restricții*

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

*Exemple*

nrdiv.in	nrdiv.out	Explicații
3	2	13 are doi divizori (1 și 13)
13	1	1 are un divizor (1)
1	8	24 are 8 divizori (1, 2, 3, 4, 6, 8, 12, 24)
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  $d_1^{p_1} d_2^{p_2} \dots d_k^{p_k}$ , atunci numărul de divizori ai lui  $x$  este  $(p_1+1)(p_2+1) \dots (p_k+1)$ .

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 educație fizică, profesorul a cerut elevilor să se alinieze. Fiind un profesor pasionat de informatică, a observat imediat că există situații în care un copil de înălțime maximă este așezat lângă un copil de înălțime minimă. Scrieți un program care să determine câte perechi de elevi alăturați îndeplinesc această condiție.

### Soluție

Vom reține înălțimile copiilor, în ordinea în care s-au aliniat aceștia, în vectorul  $h$ . Determinăm mai întâi înălțimea celui mai înalt copil și înălțimea celui mai scund copil. Parcurgem apoi vectorul, identificând toate perechile de copii alăturați în care primul copil are înălțimea maximă și al doilea, înălțimea 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 realinieze 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  $a \leq b$ , 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 ( $s < 1000000000$ ). 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  $s \cdot (s-1) / 2$  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