ALGORITMI CU VECTORI (1)

Declararea **statică** a unui vector:

`float v[50];` //spatiul maxim rezervat: v[0], v[1], …, v[49], elemente reale

`int n;` //numarul efectiv de elemente din vector, ele ocupand pozitiile 0,1,…, *n*-1

Accesarea unui element din vector se face cu v[i], unde i este pozitia in vector.

Alocarea **dinamică** a unui vector (pentru avansați):

`int n;` //numarul efectiv de elemente din vector, ele ocupand pozitiile 0,1,…, *n*-1

//cunoscut, de exemplu cu `cin>>n;`

`float *v;` //v este pointer catre un real sau mai multi (el memoreaza adresa de inceput a variabilei / variabilelor la care se refera)

`v = new float [n];` //se rezerva spatiu in memorie pentru n valori reale

Accesarea unui element din vector se face tot cu v[i].

Dupa utilizarea vectorului se elibereaza memoria ocupata de elementele la care se refera pointerul:

`delete [] v;` //operator tot din C++ ca si new

R1 (2*).

Enunțul problemei: Să se determine suma elementelor unui vector.

Metoda de rezolvare: De exemplu, pentru vectorul $v = (1\ 3\ -2\ 3.2)$ suma elementelor este 5.2. Pentru un vector $v = (v_1, v_2, ..., v_n)$ se procedează ca în cazul general pentru determinarea unei sume, ținând cont ca această sumă se scrie restrâns: $S = \sum_{i=1}^{n} v_i$.

O descriere a algoritmului în pseudocod:

```
citeşte n,v1,…,vn
S ← 0                    * suma se initializeaza cu 0
pentru i = 1,n,1 repeta     *parcurgem vectorul
      S ← S + vi         *la suma anterioara adaug elem.curent
scrie S                  *afisam valoarea finala a variabilei S
```

O descriere a algoritmului în C++ (CodeBlocks):

În primul rând, va apare un decalaj în ceea ce priveşte indicii din şir:

Pseudocod	1	2	…	*n* - 1	*n*
C / C++	v_0	v_1	…	v_{n-2}	v_{n-1}

```
#include <iostream>
using namespace std;
int main() {
 int n,i;
 float v[30], S; //maxim v[0], v[1] … v[29]
 //intai citim nr efectiv de elem si apoi elementele vect
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 for (i=0;i<n;i++)
  {
   cout<<"v["<<i+1<<"]=";
   //val nr. i+1 este memorata in vector pe pozitia i
   cin>>v[i];
  }
 //calculam suma elementelor vectorului
 S = 0;                  //suma se initializeaza cu 0
 for (i=0;i<n;i++)       //parcurgem vector
   S += v[i];    //la S se adauga valoarea curenta din vector
```

```
 cout<<"Suma elementelor vectorului dat este "<<S<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea vectorului: 4 <Enter>
v[1] = 1 <Enter>
v[2] = 3 <Enter>
v[3] = -2 <Enter>
v[4] = 3.2 <Enter>
Suma elementelor vectorului dat este 5.2
```

sau se poate aduna la sumă elementul vectorului imediat ce este citit de la tastatură

```
#include <iostream>
using namespace std;
int main()
{int n,i;
 float v[30],S = 0; //maxim v[0], v[1] … v[29]
 //intai citim nr efectiv de elem si apoi elementele vect
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele: ";
 for (i=0;i<n;i++)
 {
  cin>>v[i]; //citesc elem. fara a mai afisa
  S += v[i]; //imediat dupa citirea componentei o adaug la suma
 }
 cout<<"Suma elementelor vectorului dat este "<<S<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea vectorului: 4 <Enter>
Dati elementele: 1 3 -2 3.2 <Enter>
Suma elementelor vectorului dat este 5.2
```

Observaţie: Pentru a aduna doar componentele cu o anumită proprietate, de exemplu cele pozitive, mai punem o condiţie înainte de a aduna componenta curentă:

```
citeşte n,v1,…,vn
S ← 0
pentru i = 1,n,1 repeta
    daca vi>0 atunci
        S ← S + vi
scrie S
```

R2 (4*) - suplimentar.

Enunţul problemei: Să se determine media aritmetică, media geometrică şi media armonică a elementelor unui vector de maxim 30 componente strict pozitive.

Metoda de rezolvare: Reamintim definiţiile mediilor pentru vectorul $(v_i)_{i=1,2,\ldots,n}$, cu $v_i>0$:

$$\text{media aritmetică} = \frac{v_1+v_2+\ldots+v_n}{n} = \frac{\sum_{i=1}^{n} v_i}{n}$$

$$\text{media geometrică} = \sqrt[n]{v_1 v_2 \ldots v_n} = \sqrt[n]{\prod_{i=1}^{n} v_i}$$

$$\text{media armonică} = \frac{n}{\frac{1}{v_1}+\frac{1}{v_2}+\ldots+\frac{1}{v_n}} = \frac{n}{\sum_{i=1}^{n}\frac{1}{v_i}}$$

Astfel, calculele au la bază determinarea a două sume şi a unui produs.

O descrierea algoritmului în pseudocod:

```
citeşte n,v1,…,vn      * vi trebuie sa fie strict pozitive
ma ← 0
pentru i = 1,n repeta
   ma ← ma + vi
ma ← ma /n
mg ← 1
pentru i = 1,n repeta
      mg ← mg * vi
mg ← ∛mg
marm ← 0
pentru i = 1,n repeta
      marm ← marm + 1/vi
marm ← n/marm
scrie ma,mg,marm
```

O descrierea algoritmului în C++:

```
#include <iostream>
#include <iomanip>
#include <math.h>
#include <stdlib.h> //pentru exit = iesire fortata din program
using namespace std;
int main() {
 int n,i;
 float v[30],ma,mg,marm;
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 for (i=0;i<n;i++)
 {
   cout<<"v["<<i+1<<"]=";
   cin>>v[i];
   if (v[i]<=0) //controlez val. i introdusa necorespunzător
    {//vi nu este strict pozitiva
     cout<<"componentele vectorului sa fie pozitive"<<endl;
     exit(1); //se iese din program
     //sau return 0; //iesire din functia principala
    }
  }
 ma = marm = 0; //orice suma se initializeaza cu 0
 mg = 1; //orice produs se initializeaza cu 1
 for (i=0;i<n;i++) //folosim un singur "for" in loc de 3
 { ma += v[i];
   mg *= v[i];
   marm += 1/v[i];
 }
 ma /= n;
 mg = pow(mg, 1.0/n);
 marm = n/marm;
 cout<<"Media aritmetica: "<<fixed<<setprecision(2)<<ma<<endl;
 cout<<"Media geometrica: "<<mg<<endl;
 cout<<"Media armonica: "<<marm<<endl;
 return 0;
}
```

R3 (2-3*).

Enunţul problemei: Să se determine valoarea minimă şi maximă dintre elementele unui vector.

Metoda de rezolvare: De exemplu, pentru elementele 3 5 2 4 7 6, valoarea minimă este 2 şi valoarea maximă este 7. Se pot iniţializa valoarea maximă şi valoarea minimă cu valoarea elementului de pe prima poziţie din vector (ca şi cum am presupune că valorea minimă şi valoarea maximă sunt egale cu valoarea vectorului de pe prima poziţie). Apoi se parcurg celelalte elemente ale vectorului şi dacă valoarea curentă este mai mică/mare decât valoarea minimă/maximă găsită până acum, se actualizează aceste valori cu valoarea curentă – la bază stă formula $\min\{v_1,..., v_i\} = \min\{v_i, \min\{v_1,..., v_{i-1}\}\}$, pentru $i \geq 2$.

O descriere a algoritmului în pseudocod:

```
citeşte n,v1,…,vn
min ← v1   *pp ca val min si maxima sunt pe prima pozitie
max ← v1
pentru i = 2, n repeta  *ne uitam la valorile a 2-a … a n-a
    daca vi<min atunci *val curenta "bate" min gasit pana acum
         min ← vi       *se actualizeaza val minima
    daca vi>max atunci *val curenta "bate" max gasit pana acum
         max ← vi       *se actualizeaza valoarea maxima
scrie min,max
```

O descriere a algoritmului în C++ (CodeBlocks):

```
#include <iostream>
using namespace std;
int main()
{
 int n,i;
 float v[30],min,max; //maxim v[0], v[1] … v[29]
 //intai citim nr efectiv de elem si elementele vect
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 for (i=0;i<n;i++)
  {
   cout<<"v["<<i+1<<"]=";
   //val. nr. i+1 este memorata in vector pe pozitia i
   cin>>v[i];
  }
 //det valoarea minima / maxima
 min = max = v[0]; //presupun ca prima val este si min si max
 for (i=1;i<n;i++) //celelalte pozitii din vector
  {
   if (v[i]<min) //daca valoarea curenta < min gasit pana acum
      min = v[i]; //actualizez valoarea minima
   if (v[i]>max) //daca valoarea curenta > max gasit pana acum
      max = v[i]; //actualizez valoarea maxima
  }
 cout<<"Valoarea minima din vector: "<<min<<endl;
 cout<<"Valoarea maxima din vector: "<<max<<endl;
 return 0;
}
```

De exemplu, pentru $n = 6$ şi $v = [3\ 5\ 2\ 4\ 7\ 6]$:

Pas	Instrucţiune	min	max
1	`min = max = v[0];`	3	3
2	`pentru i=1: v[i]=5≮min=3, v[i]=5>max=3`	3	5
3	`pentru i=2: v[i]=2<min=3, v[i]=2≯max=5`	2	5

4	pentru i=3: v[i]=4≮min=2, v[i]=4≯max=5	2	5
5	pentru i=4: v[i]=7≮min=2, v[i]=7>max=5	2	7
6	pentru i=5: v[i]=6≮min=2, v[i]=6≯max=7	2	7

Observaţie: O altă variantă este aceea în care nu se face iniţializarea valorilor minime şi maxime cu prima valoare din vector (este cazul algortimilor în care se cere determinarea valorii minime/maxime dintre elementele unui vector ce îndeplinesc o anumită proprietate – aici nu ştim exact dacă prima componentă îndeplineşte sau nu acea proprietate). Dacă valorile vectorului aparţin unui domeniu de valori, adică $v_i \in [m, M]$ pentru orice i, atunci se poate iniţializa minimul cu M şi maximul cu m, apoi prima componentă va fi mai mică / mare decât valoarea minimă /maximă şi se vor actualiza valorile minime/maxime de până cum.

O altă variantă a algorimului curent este aceea în care nu se folosesc vectori căci nu am nevoie neapărat să memorez valorile, ci le citim pe rând într-o singură variabilă. De asemenea, aici am presupus că se folosesc valori întregi:

```
#include <iostream>
//#include <cmath> //pt INFINITY
#include <climits> //pt  LONG_MIN=-2^31+1/LONG_MAX=2^31
using namespace std;
int main()
{int a,min,max,n,i; //min si max trebuie sa fie de acelasi
 //tip de date ca valorile citite

 //min = INFINITY; //pp ca valoarea min este infinit
 //max = -INFINITY; //pp ca valoarea min este -infinit
 min = LONG_MAX; //pp ca valoarea min este 2147483647
 max = -MAXINT-1; //pp ca valoarea min este -2147483648
 cout<<"Cate elemente introduceti? "; cin>>n;
 cout<<"Dati elementele: ";
 for (i=0;i<n;i++) //pentru fiecare dintre elem. vect.
  {
   cin>>a;            //citim valoarea in aceeasi variabila a
   if (a<min) min=a;
   if (a>max) max=a;
  }
 cout<<"Valoarea minima: "<<min<<endl;
 cout<<"Valoarea maxima: "<<max<<endl;
 return 0;
}
```

Macro constants <climits.h>

name	expresses	value*
CHAR_BIT	Number of bits in a char object (byte)	8 or greater*
SCHAR_MIN	Minimum value for an object of type signed char	-127 (-2^7+1) or less*
SCHAR_MAX	Maximum value for an object of type signed char	127 (2^7-1) or greater*
UCHAR_MAX	Maximum value for an object of type unsigned char	255 (2^8-1) or greater*
CHAR_MIN	Minimum value for an object of type char	either SCHAR_MIN or 0
CHAR_MAX	Maximum value for an object of type char	either SCHAR_MAX or UCHAR_MAX
MB_LEN_MAX	Maximum number of bytes in a multibyte character, for any locale	1 or greater*
SHRT_MIN	Minimum value for an object of type short int	-32767 ($-2^{15}+1$) or less*
SHRT_MAX	Maximum value for an object of type short int	32767 ($2^{15}-1$) or greater*
USHRT_MAX	Maximum value for an object of type unsigned short int	65535 ($2^{16}-1$) or greater*

INT_MIN	Minimum value for an object of type int	-32767 ($-2^{15}+1$) or less*
INT_MAX	Maximum value for an object of type int	32767 ($2^{15}-1$) or greater*
UINT_MAX	Maximum value for an object of type unsigned int	65535 ($2^{16}-1$) or greater*
LONG_MIN	Minimum value for an object of type long int	-2147483647 ($-2^{31}+1$) or less*
LONG_MAX	Maximum value for an object of type long int	2147483647 ($2^{31}-1$) or greater*
ULONG_MAX	Maximum value for an object of type unsigned long int	4294967295 ($2^{32}-1$) or greater*
LLONG_MIN	Minimum value for an object of type long long int	-9223372036854775807 ($-2^{63}+1$) or less*
LLONG_MAX	Maximum value for an object of type long long int	9223372036854775807 ($2^{63}-1$) or greater*
ULLONG_MAX	Maximum value for an object of type unsigned long long int	18446744073709551615 ($2^{64}-1$) or greater*

R4 (2-3*).

Enunțul problemei: Să se determine câte numere pozitive, negative, respectiv nule sunt într-un vector de *n* numere reale dat.

Metoda de rezolvare: Algoritmul constă în citirea dimensiunii vectorului şi apoi a componentelor valorilor, după care se analizează fiecare valoare a vectorului: dacă este pozitivă aceasta se va contoriza la elementele pozitive, dacă este negativă aceasta se va contoriza la elementele negative, respectiv dacă este zero aceasta se va contoriza la elementele nule. Analiza componentelor se poate face după ce se citesc efectiv toate elementele vectorului sau se poate face imediat după citire evitându-se astfel o a doua parcurgere a vectorului. Mai jos, în descrierea algoritmului în pseudocod analiza componentelor se face dupa citirea tuturor componentelor, dar în descrierea algoritmului în C++ analiza se face imediat după citirea fiecărei componente.

O descriere a algoritmului în pseudocod:

```
citeşte n,v1,...,vn *citim dimensiunea si elementele vectorului
poz ← 0 *contorul numerelor pozitive se initializeaza cu 0
neg ← 0 *contorul numerelor pozitive se initializeaza cu 0
nule ← 0 *contorul zerourilor se initializeaza cu 0
pentru i = 1,n repeta          *parcurgem vectorul
  daca vi > 0 atunci           *daca curenta este pozitiva
     poz ← poz + 1             *se numara la contorul pozitivelor
  altfel *vi≥0
    daca vi < 0 atunci         *altfel daca curenta este pozitiva
      neg ← neg + 1            *se numara la contorul negativelor
    altfel
      nule ← nule + 1          *se numara la contorul zerourilor
scrie poz, neg, nule           *se afiseaza valorile contoarelor
```

O descriere a algoritmului în C++:

```
# include <iostream>
using namespace std;
int main()
{
 float v[20];                  //disponibile: x[0], ..., x[19]
 int n,nule,neg,poz;           //ocupate: x[0], ..., x[n-1]
 nule = poz = neg = 0;         //initializam contoarele
 //datele de intrare = dimensiunea si elementele vectorului
 cout<<"Dati dim. vect: "; cin>>n;
 cout<<"Dati elementele vectorului:"<<endl;
```

```
  for (int i=0; i<n; i++) {
    cout<<"v["<<i+1<<"]=";
    cin>>v[i];                //imediat dupa ce-am citit
    if (v[i]>0) poz++;        //comparam valoarea cu 0
    else
       if (v[i]<0) neg++;
       else nule++;
   }
  //afisam rezultatele
  cout<<"Numarul valorilor pozitive: "<<poz<<endl;
  cout<<"Numarul valorilor negative: "<<neg<<endl;
  cout<<"Numarul valorilor nule: "<<nule<<endl;
  return 0;
}
```

R5 (3-4*).

Enunţul problemei: Fie $v = (v_1, \ldots, v_n)$. Să se calculeze media vectorului $m = \frac{1}{n}\sum_{i=1}^{n} v_i$ şi dispersia vectorului $s = \frac{1}{n}\sum_{i=1}^{n} (v_i - m)^2$. De exemplu, pentru v = (1 2 3 6): m = (1 +2+3+6) / 4 = 12 / 4 = 3, iar $s = \frac{1}{4}\left((1-3)^2 + (2-3)^2 + (3-3)^2 + (6-3)^2\right) = \frac{14}{4}$ = 3.5.

Metoda de rezolvare: Media vectorului reprezintă chiar media aritmetică a componentelor vectorului (adică suma componentelor împărţită la numărul componentelor) , iar dispersia reprezintă de asemenea o sumă împărţită la numărul componentelor. Întâi calculăm suma componentelor în variabila *s*, apoi împărţim rezultatul din *s* la *n* şi actualizăm *m*. Similar pentru *s*, întâi calculăm suma pătratelor diferenţei dintre valoarea curentă şi *m* în variabila *s*, apoi împărţim rezultatul din *s* la *n* şi actualizăm *s*.

O descriere a algoritmului în pseudocod:

```
citeşte n,v1,...,vn *citim dimensiunea si elementele vectorului
m ← 0               *media vectorului
pentru i = 1,n repeta      *parcurgem vectorul
   m ← m + vi         *adaugam la m componenta curenta
m ← m / n             *se imparte rezultatul la n
s ← 0                 *dispersia vectorului
pentru i = 1,n repeta      *parcurgem vectorul
   s ← s + (vi-μ)2    *adaugam la s componenta curenta
s ← s / n             *se imparte rezultatul la n
scrie m, s
```

O descriere a algoritmului în C++:

```
# include <iostream>
# include <iomanip>
using namespace std;
int main() {
 int n,i;
 float v[15],m,s;
 //se citesc dimensiunea si elementele vectorului
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele vectorului:"<<endl;
```

```
 for (i=0;i<n;i++) {
    cout<<"v["<<i+1<<"]=";
    cin>>v[i];
   }
 //calculam m
 m = 0;
 for (i=0;i<n;i++) m += v[i];    //suma componentelor vect.
 m /= n;                         //impartita la n
 //calculam sigma
 s = 0;
 for (i=0;i<n;i++) s += (v[i]-m)*(v[i]-m);
 s /= n;

 //afisam valorile obtinute
 cout<<"Media vectorului = "<<fixed<<setprecision(2)<<m<<endl;
 cout <<"Dispersia vectorului = "<<s<<endl;
 return 0;
}
```

R6 (3*).

Enunţul problemei: Fie vectorii $u = (u_1, \ldots, u_n)$ şi $v = (v_1, \ldots, v_n)$. Să se calculeze produsul scalar dintre cei doi vectori şi normele ambilor vectori. De exemplu, pentru u = (-1 1 2 0) şi v = (1 2 3 6), produsul scalar dintre cei doi vectori este $< u, v > = (-1)\cdot 1 + 1\cdot 2 + 2\cdot 3 + 0\cdot 6 = 7$, norma vectorului u este $||u|| = \sqrt{(-1)^2 + 1^2 + 2^2 + 0^2} = \sqrt{6} \sim 2.45$, norma vectorului v este $||v|| = \sqrt{1^2 + 2^2 + 3^2 + 6^2} = \sqrt{50} \sim 7.07$.

Metoda de rezolvare: Pentru $u = (u_1, \ldots, u_n)$ şi $v = (v_1, \ldots, v_n)$:

- produsul scalar dintre cei doi vectori este $<u,v> = \sum_{i=1}^{n} u_i v_i$
- norma vectorului u este $||u|| = \sqrt{\sum_{i=1}^{n} (u_i)^2}$
- norma vectorului v este $||v|| = \sqrt{\sum_{i=1}^{n} (v_i)^2}$.

Aşadar, produsul scalar este o sumă, iar normele pot fi văzute iniţial ca fiind sumele de sub radicali, apoi se corectează prin calcularea radicalului din sumele calculate anterior.

O descriere a algoritmului în pseudocod:

```
citeşte n,u1,...,un,v1,...,vn
*am citit dimensiunea comuna si elementele celor 2 vectori
ps ← 0              *media vectorului
pentru i = 1,n  repeta      *parcurgem vectorul
   ps ← ps + ui·vi          *adaugam la miu componenta curenta

Nu ← 0                      *norma vectorului u, initial 0
pentru i = 1,n  repeta      *parcurgem vectorul u
   Nu ← Nu + (ui)^2         *se adauga patratul compon.curente
Nu ← √Nu             *se corecteaza valoarea variabilei Nu
Nv ← 0                      *norma vectorului u, initial 0
pentru i = 1,n  repeta      *parcurgem vectorul u
   Nv ← Nv + (vi)^2         *se adauga patratul compon.curente
Nv ← √Nv             *se corecteaza valoarea variabilei Nv
scrie ps, Nu, Nv
```

O descriere a algoritmului în C++ (sub CodeBlocks):

```
# include <iostream>
# include <iomanip>
# include <math.h>
using namespace std;
int main() {
 int n,i;
 float u[20],v[20],ps=0,Nu=0,Nv=0;
 //se citesc dimensiunea comuna si elementele vectorilor
 cout<<"Dati dimensiunea vectorilor: "; cin>>n;
 cout<<"Dati elementele vectorului u:"<<endl;
 for (i=0;i<n;i++)
  { cout<<"u["<<i+1<<"]="; cin>>u[i]; }
 cout<<"Dati elementele vectorului v:"<<endl;
 for (i=0;i<n;i++)
  { cout<<"v["<<i+1<<"]="; cin>>v[i]; }
for (i=0;i<n;i++) {          //cu un singur "for" se calculeaza:
   ps += u[i]*v[i];          //suma din u[i]*v[i]
   Nu += u[i]*u[i];          //suma din u[i]*u[i]
   Nv += v[i]*v[i];          //suma din v[i]*v[i]
  }
 Nu = sqrt(Nu);             //se corecteaza valoarea var.Nu
 Nv = sqrt(Nv);             //se corecteaza valoarea var.Nu
 //afisam valorile obtinute
 cout<<"Produsul scalar dintre u si v = "<<fixed<<
setprecision(2)<<ps<<endl;
 cout<<"Norma vectorului u = "<<Nu<<endl;
 cout<<"Norma vectorului v = "<<Nv<<endl;
 return 0;
}
```

O altă implementare în C++ a acestui algoritm poate folosi funcţie pentru calculul produsului scalar şi aceasta foloseşte şi la calculul normei unui vector ţinând cont că norma unui vector se poate exprima în funcţie de produsul scalar dintre doi vectori egali ($||u|| = \sqrt{<u,u>}$):

```
# include <iostream>
# include <iomanip>
# include <math.h>
using namespace std;
float ps(float u[20], float v[20], int n)
{ //functie ce calculeaza produsul scalar dintre vectorii
  //dati ca primii do parametri ai functiei, ambii de dim. n
 float s = 0;
 for (int i=0;i<n;i++) s += u[i]*v[i];
 return s;
}
int main() {
 int n,i;
 float u[20],v[20];
 //se citesc dimensiunea comuna si elementele vectorilor
 cout<<"Dati dimensiunea vectorilor: "; cin>>n;
 cout<<"Dati elementele vectorului u:"<<endl;
 for (i=0;i<n;i++)
  { cout<<"u["<<i+1<<"]="; cin>>u[i]; }
 cout<<"Dati elementele vectorului v:"<<endl;
 for (i=0;i<n;i++)
```

```
  { cout<<"v["<<i+1<<"]="; cin>>v[i]; }
 //afisam valorile obtinute
 cout<<"Produsul scalar dintre u si v = " << fixed<<
setprecision(2) << ps(u,v,n)<<endl;
 cout<<"Norma vectorului u = "<<sqrt(ps(u,u,n))<<endl;
 cout<<"Norma vectorului v = "<<sqrt(ps(v,v,n))<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea vectorilor: 4 <Enter>
Dati elementele vectorului u:
u[1] = -1 <Enter>
u[2] = 1 <Enter>
u[3] = 2 <Enter>
u[4] = 0 <Enter>
Dati elementele vectorului v:
v[1] = 1 <Enter>
v[2] = 2 <Enter>
v[3] = 3 <Enter>
v[4] = 6 <Enter>
Produsul scalar dintre u si v = 7
Norma vectorului u = 2.45
Norma vectorului v = 7.07
```

R7 (3*).

Scrieţi un algoritm pentru a stabili dacă un vector are toate componentele pare (v_i par, pentru orice $i = 0,\ldots, n\text{-}1$) sau nu.

```
#include <iostream>
using namespace std;

void Citire(int v[20], int &n)
{//functie care citeste numarul efectiv de elemente din vector
 //si valorile componentelor vectorului
 cout<<"Dati numarul efectiv de elemente: "; cin>>n;
 cout<<"Dati elementele vectorului: ";
 for (int i=0;i<n;i++) cin >> v[i];
}

bool ToatePare(int v[20], int n)
{//se testeaza daca toate elem. din vector sunt pare sau nu
 for (int i=0;i<n;i++)
  if (v[i]%2) //==1 <=> componenta curenta este impara
       return false; //functia returneaza valoarea false
 return true; //nu s-a iesit pana acum din functie cu false
}

int main()
{int v[20], n; //v[0],v[1],…,v[19] = intregi, pare sau nu
 Citire(v,n);
 if (ToatePare(v,n))  //==true
   cout<<"Vectorul are toate componentele pare"<<endl;
 else
   cout<<"Vectorul nu are toate componentele pare"<<endl;
 return 0;
}
```

R8 (3*).

Scrieţi un algoritm pentru a stabili dacă un vector de numere reale este crescător ($v_i \le v_{i+1}$, pentru orice i) sau nu.

Metoda de rezolvare: O metodă de descriere a algoritmului ar fi aceea în care la început se presupune că vectorul $v = (v_1, v_2, \ldots, v_n)$ este crescător, apoi se parcurge vectorul cu $i = 1,\ldots, n\text{-}1$ şi în cazul în care $v_i \not\le v_{i+1}$, atunci nu este crescător. În C/C++ apare decalajul de indici: pentru vectorul `v[0], …, v[n-1]`, se parcurge vectorul cu `i=0,…,n-2` şi în cazul în care două componente vecine nu sunt corespunzătoare, adică `v[i] > v[i+1]`, atunci nu este crescător (pentru aceasta putem folosi o variabilă care va avea valoarea 1 (sau true) în cazul în care vectorul este crescător şi valoarea 0 (sau false) în cazul în care vectorul nu este crescător).

O descriere a algoritmului în pseudocod:

```
citeşte n,v1,...,vn *citim dimensiunea si elementele vectorului

crescator ← adevarat        *presupunem ca vect. este crescator
pentru i = 1,n-1  repeta    *parcurgem vectorul
   daca v[i]>v[i+1] atunci  *daca doua elemente vecine
      crescator ← fals      *atunci vectorul nu este crescator

daca crescator = adevarat atunci      *testam valoarea finala
     scrie "Este vector crescator"
altfel
     scrie "Nu este vector crescator"
```

O descriere a algoritmului în C++:

```
#include <iostream>
using namespace std;

int main() {
 float v[20]; //componentele pot fi si reale
 int n,i;
 bool crescator;

 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele vectorului: "<<endl;
 for (i=0;i<n;i++)
 { cout<<"v["<<i+1<<"]="; cin >> v[i]; }

 crescator = true; // Presupunem ca vectorul este crescator
 for (i=0;i<=n-2;i++)
  if (v[i]>v[i+1]) //daca doua dintre componentele vecine
    {              //sunt "pe dos"
     crescator = false; //este clar: sirul nu este crescator
     break;          //iesire din for (nu merg mai departe)
    }

 if (crescator) //sau ==true
     cout<<"Sirul este crescator"<<endl;
 else cout<<"Sirul nu este crescator"<<endl;
 return 0;
}
```

sau folosind funcţii:

```
#include <iostream>
using namespace std;

void Citire(float v[20], int &n)
{
 cout<<"Dati numarul efectiv de elemente: "; cin>>n;
 cout<<"Dati elementele vectorului: ";
```

```
 for (int i=0;i<n;i++) cin >> v[i];
}
bool Crescator(float v[20], int n) {
 for (int i=0;i<=n-2;i++)
  if (v[i]>v[i+1]) // sau (!(v[i]<=v[i+1])
      return false;
 return true;
}
int main()
{
 float v[20]; //elementele pot fi reale
 int n;

 Citire(v,n);
 if (Crescator (v,n))  //==true
   cout<<"Sirul este crescator"<<endl;
 else
   cout<<"Sirul nu este crescator"<<endl;
 return 0;
}
```

Observaţie: Dacă se cere determinarea faptului dacă vectorul este **monoton sau nu** se poate întâi testa dacă este monoton crescător; în caz afirmativ se afişează un mesaj, altfel se testează similar dacă vectorul este monoton descrescător.

```
#include <iostream>
using namespace std;
int main()
{
 int n,i;
 float v[20];
 bool mon_cresc,mon_descresc;
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele vectorului: "<<endl;
 for (i=0;i<n;i++)
 { cout<<"v["<<i+1<<"]="; cin >> v[i]; }
 mon_cresc = true; // Presupunem ca vectorul este crescator
 for (i=0;i<=n-2;i++)
  if (v[i]>v[i+1]) { //elementele sunt "pe dos"
    mon_cresc = false;
    break; //nu se merge mai departe (se iese din for)
   }
 if (mon_cresc) //sau if (mon_cresc==true)
   cout<<"Sirul este crescator => monoton"<<endl;
 else {
   mon_descresc = true; // pp. ca vectorul este descrescator
   for (i=0;i<=n-2;i++) //componentele care au element urmator
     if (v[i]<v[i+1]) { //elementele sunt "pe dos"
         mon_descresc=false;
         break;
       }
   if (mon_descresc) //sau if (mon_descresc==true)
      cout<<"Sirul este descrescator => monoton"<<endl;
   else cout<<"Sirul nu este monoton"<<endl;
  }
 return 0;
}
```

sau folosind funcţii:

```
#include <iostream>
using namespace std;

void Citire(float v[20], int &n)
{
 cout<<"n="; cin>>n;
 cout<<"Dati elementele vectorului: ";
 for (int i=0;i<n;i++) cin >> v[i];
}

bool Crescator(float v[20], int n)
{
 for (int i=0;i<=n-2;i++)
  if (v[i]>v[i+1]) // sau (!(v[i]<=v[i+1])
      return false;
 return true;
}

bool Descrescator(float v[20], int n) {
 for (int i=0;i<=n-2;i++)
  if (v[i]<v[i+1]) // sau (!(v[i]>=v[i+1])
      return false;
 return true;
}

int main() {
 float v[20];
 int n;

 Citire(v,n);
 if (Crescator (v,n) || Descrescator (v,n))
   cout<<"Sirul este monoton"<<endl;
 else
   cout<<"Sirul nu este monoton"<<endl;
 return 0;
}
```

R9 (3-4*).

Scrieţi un algoritm pentru a stabili dacă un vector de numere reale are componentele în progresie aritmetică sau nu.

Metoda de rezolvare: Un vector $v = (v_1, v_2, \ldots, v_n)$ are elementele în progresia aritmetică dacă ($v_i = \frac{v_{i-1} + v_{i+1}}{2}$ pentru orice $i = 2,\ldots, n-1$ – acele elemente care au vecini în stânga şi în dreapta). O metodă de descriere a algoritmului ar fi aceea în care la început se presupune că vectorul $v = (v_1, v_2, \ldots, v_n)$ are elementele în progresie aritmetică, apoi se parcurge vectorul cu $i = 2,\ldots, n-1$ şi în cazul în care $v_i \neq \frac{v_{i-1} + v_{i+1}}{2}$, atunci nu are elemntele în progresie aritmetică.

În C/C++ apare decalajul de indici: pentru vectorul `v[0], …, v[n-1]`, se parcurge vectorul cu `i=1,…,n-2` şi în cazul în care v[i] !=(v[i-1]+v[i+1])/2, atunci vectorul nu are elementele în proresie aritmetică (pentru aceasta putem folosi o variabilă care va avea valoarea true în cazul în care vectorul are elementele în progresie aritmetică şi valoarea false în cazul în care vectorul nu are elementele în progresie aritmetică).

```
#include <iostream>
using namespace std;
int main()
{
 int n,i;
 float v[20];
 bool p_a;
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele vectorului: "<<endl;
 for (i=0;i<n;i++)
 {cout<<"v["<<i+1<<"]=";
  cin >> v[i];
 }
 p_a = true; // Presupunem ca vectorul este crescator
 for (i=1;i<=n-2;i++) //componentele care au vecini
  if (v[i] != (v[i-1]+v[i+1])/2) //elementele sunt "pe dos"
   {
    p_a = false; //este clar ca nu este progresie aritmetica
    break;       //se iese din for (nu mergem mai departe)
   }
 if (p_a) //sau if (p_a==true)
     cout<<"Este progresie aritmetica"<<endl;
 else cout<<"Nu este progresie aritmetica"<<endl;
 return 0;
}
```

sau nefolosind variabila p_a:

```
#include <iostream>
using namespace std;
int main() {
 int n,i; float v[20];
 cout<<"n="; cin>>n;
 cout<<"Dati elementele vectorului: "<<endl;
 for (i=0;i<n;i++) {cout<<"v["<<i+1<<"]=";cin >> v[i]; }
 for (i=1;i<=n-2 && v[i] == (v[i-1]+v[i+1])/2; i++) ;
 if (i==n-1) //toate pana la n-2 au fost OK
     cout<<"Este progresie aritmetica"<<endl;
 else cout<<"Nu este progresie aritmetica"<<endl;
 return 0;
}
```

sau folosind funcţii

```
#include <iostream>
using namespace std;
void Citire(float v[20], int &n)
{cout<<"n="; cin>>n;
 cout<<"Dati elementele vectorului: ";
 for (int i=0;i<n;i++) cin >> v[i];
}
bool EsteProgresieAritmetica(float v[20], int n)
{for (int i=1; i<=n-2; i++) //componentele care au vecini
   if (v[i] != (v[i-1]+v[i+1])/2) //tripletul nu este in p.a.
     return false; //este clar ca nu este progresie aritmetica
  return true; //n-am iesit cu return false, deci este OK
}
int main() {
 int n;
 float v[20];
 Citire(v,n);
 if (EsteProgresieAritmetica(v,n)) //==true
```

```
        cout<<"Este progresie aritmetica"<<endl;
 else cout<<"Nu este progresie aritmetica"<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea vectorului: 5 <Enter>
Dati elementele vectorului:
v[1] = 3 <Enter>
v[2] = 6 <Enter>
v[3] = 9 <Enter>
v[4] = 12 <Enter>
v[5] = 15 <Enter>
Este progresie aritmetica
```

R10 (2*).

Enunţul problemei: Se consideră rezultatele obţinute la examenul de Algoritmi şi structuri de date de studenţii de la matematică-informatică, anul 1. Să se descrie un algoritm pentru a determina procentul studenţilor promovaţi. De exemplu, pentru 4 note: 9 5 4 7, procentul de promovabilitate este de 75%.

Metoda de rezolvare: Rezultatele se pot reţine într-un vector, iar procentul se determină după formula *contor* * 100 / *n* (valoare între 0 şi 100), unde *contor* reprezintă numărul studenților promovați, iar *n* este numărul total de note. Înainte se parcurg elementele vectorului cu note, de dimensiune *n* şi în *contor* se numără valorile ≥ 5 din vector.

O descriere a algoritmului în C++ (CodeBlocks):

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
 int n,v[50];
 cout<<"Dati numarul de note: "; cin>>n;
 for (int i=0;i<n;i++){
    cout<<"Dati nota nr. "<<i+1<<": ";
    cin>>v[i];
 }
 int contor = 0;
 for (int i=0;i<n;i++)
    if (v[i]>=5) contor++;
 cout<<"Procentul   promovatilor   =   "<<   setprecision(2)
<< contor*100.0/n <<"%"<<endl;
 return 0;
}
```

sau, folosind funcții

```
#include<iostream>
#include<iomanip>
using namespace std;
void Citire(int v[50], int &n)
{//se citeste dimensiunea si elementele vectorului
 //pentru ca n este variabial simpla (tip int)
 //si in interiorul functiei n isi modifica val. (se citeste)
 //atunci parametrul n trebuie transferat prin parametru (&n)
 //la vector, numele vectroului = adresa primului element
 //v = &v[0] => transfer prin referinta oricum
 cout<<"Dati numarul de note: "; cin>>n;
 for (int i=0;i<n;i++)
```

```
  {
    cout<<"Dati nota nr. "<<i+1<<": ";
    cin>>v[i];
   }
}
int main()
{
 int n,v[50];
 Citire(v,n);
 int contor = 0;
 for (int i=0;i<n;i++)
    if (v[i]>=5) contor++;
 cout<<"Procentul     promovatilor     =     "<<setprecision(2)
<<contor*100.0/n<<"%"<<endl;
 return 0;
}
```

R11 (3*).

Enunţul problemei: Să se înlocuiască fiecare element al unui vector cu media aritmetică a celorlalte *n*-1 elemente din vector. De exemplu, pentru vectorul: 2 4 3 7 4 1, se obține (4+3+7+4+1)/5 (2+3+7+4+1)/5 (2+4+7+4+1)/5 (2+4+3+4+1)/5 (2+4+3+7+1)/5 (2+4+3+7+4)/5, adică vectorul (3.8 3.4 3.6 2.8 3.4 4).

Metoda de rezolvare: Se poate calcula suma *s* a componentelor inițiale ale vectorului. Apoi media aritmetic a celorlalte componente pentru poziția *i* este (*s* - *v*[*i*]) / (*n*-1).

O descriere a algoritmului în C++ (CodeBlocks):

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{int n,i;
 float v[50];
 cout<<"Dati numarul de elemente: "; cin>>n;
 for (int i=0;i<n;i++){
    cout<<"Dati elementul nr. "<<i+1<<": ";
    cin>>v[i];
   }
 //calculam suma tuturor elementelor initiale din vector
 float s = 0;
 for (i=0;i<n;i++) s += v[i];
 //acum inlocuim fiecare element cu media celorlalte elemente =
 //(suma initiala s - v[i])/(n-1)
 for (i=0;i<n;i++) v[i] = (s-v[i])/(n-1);
 //afisarea vectorului modificat
 cout<<"Dupa inlocuirea fiecarui element din vector cu media
aritmetica a celorlalte elemente:"<<endl;
 for (i=0;i<n;i++)    cout<<setprecision(2)<<v[i]<<" ";
  cout<<endl;
 return 0;
}
```

sau, folosind funcții

```
#include<iostream>
#include<iomanip>
using namespace std;

void Citire(float v[50], int &n)
{//pentru ca n este variabial simpla (tip int)
```

```
 //si in interiorul functiei n isi modifica valoarea (se citeste)
 //atunci parametrul n trebuie transferat prin parametru (&n)
cout<<"Dati numarul de elemente: "; cin>>n;
 for (int i=0;i<n;i++){
    cout<<"Dati elementul nr. "<<i+1<<": ";
    cin>>v[i];
   }
}
void Afisare(float v[50], int n)
{//parametrul n este fara &, caci nu-si modifica valoarea in functie
 for (int i=0;i<n;i++) cout<<v[i]<<" ";
 cout<<endl; }

int main()
{
 int n;
 float v[50];
 Citire(v,n);
 //calculam suma tuturor elementelor initiale din vector
 float s = 0;
 for (int i=0;i<n;i++) s += v[i];
 //acum inlocuim fiecare element cu media celorlalte elemente =
 //(suma initiala s - v[i])/(n-1)
 for (int i=0;i<n;i++) v[i] = (s-v[i])/(n-1);
 //afisarea vectorului modificat
 cout<<"Dupa inlocuirea fiecarui element din vector cu media
aritmetica a celorlalte elemente:"<<endl;
 Afisare(v,n);
 return 0;
}
```

R12 (suplimentar, 6*).

Enunţul problemei: Să se calculeze valoarea unui polinom într-un punct dat.

Metoda de rezolvare: De exemplu, pentru polinomul $P = 3X^2 - 5X + 2$ şi $x_0 = 2$ atunci $P(x_0) = P(2) = 3*2^2 - 5*2 + 2 = 4$.

În general, putem considera un polinom de forma:

$$P = a_n \cdot X^n + a_{n-1} \cdot X^{n-1} + \ldots + a_1 X + a_0$$

ce are gradul n ($a_n \neq 0$) şi coeficienţii $a_n, a_{n-1}, \ldots, a_1, a_0$.

Se cere să calculăm $P(x_0)$ pentru x_0 citit de la tastatură. O variantă ar fi cea clasică

$$P(x_0) = a_n \cdot (x_0)^n + a_{n-1} \cdot (x_0)^{n-1} + \ldots + a_1 x_0 + a_0 = \sum_{i=0}^{n} a_i (x_0)^i$$

Pentru a evita expresia 0^0 (în C++, 0^0 calculat cu funcţia pow dă rezultatul 1, însă va da un avertisment la copilare) din $\sum_{i=0}^{n} a_i (x_0)^i$ (când x_0=0 şi i=0) vom scoate din sumă termenul i=0

$$\Rightarrow P(x_0) = a_0 + \sum_{i=1}^{n} a_i (x_0)^i$$

Descrierea algoritmului în pseudocod:

```
citeşte n,an,…,a0,x0
S ← a0      *punem deja in suma termenul liber al polinomului
pentru i = n,1,-1 repeta   *parcugem coef. de la cel dominant
      S ← S + ai(x0)^i
scrie S
```

Descrierea algoritmului în pseudocod C++:

```
#include <iostream>
#include <math.h>
using namespace std;
float a[20],x0,S;
int n,i;
int main()
{
 cout<<"Dati gradul polinomului: "; cin>>n;
 cout<<"Dati coeficientii polin (de la coef cel mai
semnificativ):"<<endl;
 for (i=n; i>=0; i--)
  { cout<<"a["<<i<<"]=";  cin>>a[i];  }
 cout<<"Dati x0: "; cin>>x0;
 S = a[0]; //initializez suma cu termenul liber
 for (i=1; i<=n; i++) S += a[i]*pow(x0,i);
 cout<<"Valoarea polinomului este: "<<S<<endl;
return 0;
}
```

Desigur, se poate (și ar fi de preferat) să se evite folosirea funcției pow, prin construirea pas cu pas a puterii.

```
#include <iostream>
using namespace std;
float a[20],x0,S,p;
int n,i;
int main(){
 cout<<"Dati gradul polinomului: "; cin>>n;
 cout<<"Dati coeficientii polin (de la coef cel mai
semnificativ):"<<endl;
 for (i=n; i>=0; i--)
  { cout<<"a["<<i<<"]=";  cin>>a[i];  }
 cout<<"Dati x0: "; cin>>x0;
 S = a[0]; //initializez suma cu termenul liber
 p = 1; //x0^i este initial 1
 for (i=1; i<=n; i++)
 {
     p *= x0;
     S += a[i]*p;
 }
 cout<<"Valoarea polinomului este: "<<S<<endl;
return 0;
}
```

O variantă iterativă, mai dificilă, dar care evită folosirea funcției putere:

$P(X) = a_n \cdot X^n + a_{n-1} \cdot X^{n-1} + \ldots + a_1 X + a_0 = a_0 + X(a_1 + X(a_2 + \ldots. + X(a_n)))$

În acest caz, algoritmul este următorul.

O descriere a algoritmului în pseudocod:

```
citeşte n,an,…,a0,x0
S ← an
pentru i = n-1,0,-1 repeta
     S ← ai + x0S
scrie S
```

O descriere a algoritmului în C++:

```
#include <iostream>
using namespace std;
float a[20],x0,S;
int n,i;
int main()
{ cout<<"Dati gradul polinomului: "; cin>>n;
 cout<<"Dati coeficientii polin (de la coef cel mai
semnificativ):"<<endl;
 for (i=n; i>=0; i--)
{  cout<<"a["<<i<<"]=";    cin>>a[i];  }
 cout<<"Dati x0: "; cin>>x0;

 S = a[n];
 for (i=n-1; i>=0; i--) S = x0*S + a[i];
 cout<<"Valoarea polinomului este: "<<S<<endl;
 return 0;
}
```

R13 (suplimentar, 3-4*).

Enunţul problemei: Să se inverseze ordinea elementelor unui vector. De exemplu, pentru *v* = (1 2 3 4 5) => *v* = (5 4 3 2 1)

Metoda de rezolvare: Pentru că se doreşte inversarea efectivă a ordinii elementelor, nu doar afişarea (când se putea parcurge vectorul de la final spre sfârşit), putem parcurge elementele până la jumătatea vectorului şi să interschimbăm elementul curent cu simetricul său faţă de jumătatea vectorului: pentru un vector memorat ca în C/C++ v[0],...,v[n-1], acest lucru presupune parcurgerea cu i=0,1,...,[n/2]-1 şi interschimbarea componentelor v[i] şi v[n-1-i] (acesta se poate realiza prin intermediul unei variabile auxiliare de acelaşi tip cu elementele vectorului: aux = v[i]; v[i] = v[n-1-i]; v[n-1-i] = aux;).

O descriere a algoritmului în C++:

```
# include <iostream>
using namespace std;
int main() {
 float v[20], aux;
 int n,i;
 //se citesc dimensiunea si elementele vectorului
 cout<<"Dati dimensiunea vectorului: "; cin>>n;
 cout<<"Dati elementele vectorului:"<<endl;
 for (i=0;i<n;i++)
   { cout<<"v["<<i+1<<"]=";  cin>>v[i]; }
 //afisam vectorul initial
 cout<<"Vectorul initial este: ";
 for (i=0;i<n;i++) cout<<v[i]<<"  ";

 //inversam ordinea elementelor
 for (i=0;i<n/2;i++)
  {aux = v[i];
   v[i] = v[n-i-1];
   v[n-i-1] = aux;
  }
 //afisam vectorul cu elementele inversate
 cout<<endl<<"Vectorul cu elementele inversate este: ";
 for (i=0;i<n;i++) cout<<v[i]<<"  ";
 cout<<endl;
 return 0;
}
```

sau folosind functii

```
#include <iostream>
using namespace std;

void Citire(float v[20], int &n)
{
 cout<<"n="; cin>>n;
 cout<<"Dati elementele vectorului: ";
 for (int i=0;i<n;i++) cin >> v[i];
}
void Afisare(float v[20], int n)
{
for (int i=0;i<n;i++)
   cout << v[i]<< " ";
}
void Inversare(float v[20], int n){
float aux;
  for (int i=0; i<n/2; i++) //prima jumatate a vectorului
  { aux = v[i];
    v[i] = v[n-1-i]; //v[n-1-i] = simetricul lui v[i]
    v[n-1-i] = aux;
  }
//sau folosind: swap(v[i],v[n-1-i]);
}
int main() {
 int n;  float v[20];
 Citire(v,n);
 Cout;;
 if (EsteProgresieAritmetica(v,n)) //==true
      cout<<"Este progresie aritmetica"<<endl;
 else
      cout<<"Nu este progresie aritmetica"<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea vectorului: 5 <Enter>
Dati elementele vectorului u:
u[1] = 1 <Enter>
u[2] = 2 <Enter>
u[3] = 3 <Enter>
u[4] = 4 <Enter>
u[4] = 4 <Enter>
Elementele vectorului initial: 1 2 3 4 5
Vectorul cu elementele inversate este: 5 4 3 2 1
```

R14 (suplimentar 4-5*).

Enunţul problemei: Să se determine dacă un număr întreg este palindrom sau nu, adică se citeşte la fel de la stânga la dreapta şi de la dreapta la stânga (numărul este identic cu răsturnatul său). De exemplu, 4357534 este palindrom.

Metoda de rezolvare: O primă variantă constă în determinarea răsturnatului numărului întreg dat şi apoi testarea condiţiei ca numărul dat să coincidă cu răsturnatul său.

O variantă mai simplă este aceea în care se reprezintă numărul ca şir de caractere şi atunci trebuie verificat dacă prima cifră coincide cu ultima, a doua cu penultima, etc., până la jumătatea şirului:

1	2	…	lungime-1	lungime
sir_1	sir_2	…	$sir_{lungime-1}$	$sir_{lungime}$

adică

sir_1 ?= sir_{lungime} (primul caracter din şir coincide cu ultimul)
sir_2 ?= $sir_{\text{lungime-1}}$ (al doilea caracter din şir coincide cu penultimul)
…….
sir_i ?= $sir_{\text{lungime} + 1 - i}$, i = 1, 2,…, lungime/2.

*Descrier*ea algoritmului în pseudocod*:

```
citeşte sir
palindrom ← adevarat
pentru i = 1,lungime_sir/2 repeta
   daca sir[i] ≠ sir[lungime_sir-i+1] atunci
      palindrom ← fals
daca palindrom = adevarat atunci
      scrie "Este palindrom"
altfel
      scrie "Nu este palindrom"
```

Descrierea algoritmului în C++:

Decalajul de indici în vector faţă de C++ este următorul:

0	1	…	lungime-2	lungime-1
sir_0	sir_1	…	$sir_{\text{lungime-2}}$	$sir_{\text{lungime-1}}$

adică

sir_0 ?= $sir_{\text{lungime-1}}$ (primul caracter din şir coincide cu ultimul)
sir_1 ?= $sir_{\text{lungime-2}}$ (al doilea caracter din şir coincide cu penultimul)
…….
sir_i ?= $sir_{\text{lungime} - 1 - i}$, i = 1, 2,…, lungime/2.

Deci:

```
#include <iostream>
#include <string.h>
using namespace std;
char sir[100];
int i;
bool palindrom;
int main()
{ cout<<"Dati numarul: "; cin>>sir;
/caracterele sirului: sir[0]....sir[strlen(sir)]-1
//sir[strlen(sir)] = '\0' (caracterul sfarsit de sir)
palindrom = true; //presup. ca este palindrom (verif.prop)
for (i=0; i<=strlen(sir)/2; i++)
  if (sir[i]!=sir[strlen(sir)-i-1]) palindrom = false;
 if (palindrom) //sau (palindrom==true)
  cout<<"Este palindrom";
 else cout<<"Nu este palindrom";
 return 0; }
```

Temă:

Pentru începători:

1) Cunoscând numărul familiilor dintr-un judeţ, valoarea veniturilor lor, precum şi valoarea "coşului minim", să descrie un algoritm pentru a stabili numărul de familii ce trăiesc sub nivelul minim de trai. De exemplu, pentru 6 familii, vectorul veniturilor (900, 750, 700, 800, 680, 600) şi coşul minim de 700 de lei, avem 2 familii ce trăiesc sub nivelul minim de trai. (Sugestie: veniturile familiilor se pot reţine într-un vector, apoi se parcurge acest vector şi se contorizează componentele care au valoarea strict mai mică decât coşul minim).
2) Fie $c_1...c_n$ valorile colesterolului a n persoane. Se considera că o persoană este sănătoasă dacă are valoarea colesterolului în intervalul 50...200. Descrieţi un algoritm pentru a determina câte persoane sunt sănătoase (Sugestie: se folosește o variabilă contor cu valoarea inițială 0, se parcurge vectorul de numere întregi şi dacă valoarea curentă este între 50 și 200 se incrementează contorul).
3) Descrieţi un algoritm pentru a determina câte numere pare, respectiv impare conţine un vector de numere întregi (Sugestie: se folosesc două variabile contor inițializate cu 0, se parcurge vectorul de numere întregi şi dacă valoarea curentă este pară se incrementează contorul celor pare, altfel se incrementează contorul celor impare).
4) Să se descrie un algoritm pentru a determina de câte ori se întâmplă ca într-un vector de numere reale să existe un element ce reprezintă produsul elementelor sale vecine. De exemplu, pentru x = (2 8 4 54 18), doar 2 elemente au proprietatea dorită ($8 = 2 \cdot 4$, $4 \neq 8 \cdot 54$ şi $54 = 4 \cdot 18$), iar pentru (2 6 5) nu este niciun element cu proprietatea dorită (Sugestie: se folosește o variabilă contor cu valoarea inițială 0, se parcurge (!!!) vectorul şi dacă valoarea curentă este produsul vecinilor se incrementează contorul; la final, dacă acel contor a rămas 0, se afișează mesajul corespunzător).
5) Se citesc un număr de temperaturi medii din anumite luni, T_1, ..., T_n, ca numere întregi. Să se afișeze cu două zecimale media celor pozitive și media celor negative. (Sugestie: se însumează și se numără temperaturile negative, respectiv cele pozitive (eventual în S1, S2, nr1, nr2) inițializate cu 0, apoi se afișează cele două valori medii float(S1)/nr1, respectiv float(S2)/nr2)
6) Se consideră rezultatele obţinute la examenul de Algoritmi şi structuri de date de studenţii de la matematică-informatică. Calculați media notelor celor promovați.
7) Afișarea elementelor prime dintr-un vector.

Pentru avansaţi:

1) Se introduce un număr par de numere reale. Să se adune câte două de pe poziții consecurive și să se afișeze sumele obținute. De exemplu, pentru n=6 numere 5 4 3 6 5 5 se afișează 9 9 10.
2) La un concurs de patinaj artistic se cunosc cele n note obţinte de un concurent. Să se calculeze punctajul final ca medie aritmetică, ştiind că nu se iau în considerare nota cea mai mică şi cea mai mare.
3) Afișarea elementelor distincte dintr-un vector. De exemplu, pentru vectorul (1 2 5 2 4 1 2) valorile distincte sunt 1, 2, 5, 4.
4) Stabiliți dacă elementele dintr-un vector sunt toate distincte sau nu.
5) Stabiliți dacă mulțimea A este inclusă în mulțimea B. Elementele unei mulțimi se memorează într-un vector.
6) Scrieți algoritmi pentru intersecției, reuniunii și diferenței a două mulțimi reprezentate cu vectori.
7) Ciurul lui Eratostene (folosind vectori)
8) Determinați coeficienții derivatei unui polinom.
9) Scrieți un algoritm care folosește schemei lui Horner pentru a determina câtul și restul împărțirii unui polinom P dat la un polinom de forma $Q = X - t$.