1. Exerciții comentate

1.1. Suma elementelor unui vector de dimensiune n.

Fie $X=(x_1,x_2,...,x_n)$. Suma elementelor este $S = x_1 + x_2 + \ldots + x_n = \sum_{i=1}^{n} x_i$.

Ştiind că adunarea numerelor reale este asociativă şi că elementul neutru pentru adunare este **0**, algoritmul poate fi descris astfel:

$S_0 = 0;$

$S_1 = S_0 + x_1 = 0 + x_1 = x_1;$

$S_2 = S_1 + x_2 = x_1 + x_2;$

...

$S_n = S_{n-1} + x_n = x_1 + ... + x_{n-1} + x_n;$

$S = S_n.$

Deoarece sumele parțiale $S_0, S_1,...,S_n$ nu interesează şi, în plus, se ocupă inutil memorie internă, însumarea se va realiza în aceeași locație de memorie, cu adresa simbolică S, în care se vor depune sumele parțiale (se va cumula câte un nou element).

$S = 0;$

$S = S + x_1 = 0 + x_1 = x_1;$

$S = S + x_2 = x_1 + x_2;$

...

$S = S + x_n = x_1 + ... + x_{n-1} + x_n;$

Algoritmul recursiv poate fi descris cu ajutorul a două formule:

- formula de start: $S = 0;$
- formula recursivă: $S = S + x(i), i=1,n.$

Elementele vectorului sunt reale şi se introduc de la tastatură.

```
Program suma_elemente_vector;
Var
  x:array[1..100] of real;
  n,i:byte;
  s:real;
Begin
  write('Dimensiunea vectorului: ');
  readln(n);
  write('Elementele vectorului: ');
  for i:=1 to n do read(x[i]);
  s:=0;
  for i:=1 to n do s:=s+x[i];
  writeln('Suma = ',s:10:2)
End.
```

1.2. Suma elementelor de rang impar ale unui vector de dimensiune n.

Fie $X=(x_1,x_2,...,x_n)$. Suma elementelor de rang impar este $S=x_1+x_3+x_5+...$ Există mai multe variante de rezolvare.

∝ *Varianta 1*. Se parcurge vectorul cu indicele pornind de la valoarea inițială **1** și crescând cu pasul **2**. Datorită particularităților instrucțiunii FOR privind pasul, se utilizează structura WHILE-DO.

```
Program suma_elemente_rang_impar_1;
Var
  x:array[1..100] of real; n,i:byte; s:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do readln(x[i]);
  s:=0; i:=1;
  while i<=n do
    Begin
      s:=s+x[i];
      i:=i+2;
    End;
  writeln('Suma = ',s:10:2)
End.
```

∝ *Varianta 2*. Se parcurge integral vectorul și se selectează elementele de rang impar, testând indicele fie prin verificarea restului împărțirii lui **i** la **2** (**i mod 2**), fie prin funcția standard ODD.

```
Program suma_elemente_rang_impar_2;
Var
  x:array[1..100] of real;  n,i:byte;  s:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do readln(x[i]);
  s:=0;
  for i:=1 to n do if odd(i) then s:=s+x[i];
  writeln('Suma = ',s:10:2)
End.
```

∝ *Varianta 3*. Variabila de ciclare ia valori între **1** și cel mai apropiat întreg față de **n/2**, iar elementele vectorului se selectează utilizând indicele **2*i-1**.

```
Program suma_elemente_rang_impar_3;
Var
  x:array[1..100] of real;  n,i:byte;  s:real;
```

```
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do readln(x[i]);
  s:=0;
  for i:=1 to round(n/2) do s:=s+x[2*i-1];
  writeln('Suma = ',s:10:2)
End.
```

1.3. Media geometrică a elementelor pozitive dintr-un vector de dimensiune n.

Fie $X=(x_1,x_2,...,x_n)$. Media geometrică a elementelor pozitive este $MEDG = \sqrt[k]{\prod_{i=1,k} x_i}$, $x_i>0$. Ştiind că înmulțirea în numere reale este asociativă şi are elementul neutru **1**, produsul $P = \prod_{i=1,k} x_i$ se calculează astfel:

! formula de start: P = 1;

! formula recursivă: P = P H x(i); i=1,k; x(i)>0.

Pentru economie de memorie internă, produsul se calculează direct în variabila MEDG. Media geometrică se poate determina numai dacă numărul elementelor pozitive (**k**) este mai mare decât 2.

Deoarece în limbajul Pascal nu există operatori (sau funcții) pentru radicali şi ridicări la putere (mai mari de ordinul doi) se utilizează formula $\sqrt[n]{a^m} = e^{(m*\ln a)/n}, a > 0$, pentru determinarea căreia există funcțiile standard EXP(x) şi LN(x).

```
Program media_geometrica;
Var
  x:array[1..100] of real;
  n,i,k:byte; medg:real;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin  write('X(',i,') = '); readln(x[i]);  end;
  medg:=1; k:=0;
  for i:=1 to n do
    if x[i] > 0 then
      begin  medg:=medg * x[i]; k:=k+1  end;
  if k > 1 then
    begin
      medg:=Exp(Ln(medg)/k);
      writeln('Media geometrica = ',medg:10:2)
    end
          else writeln('Nu se poate calcula !');
End.
```

1.4. Determinarea poziției primei apariții a unei valori date într-un vector neordonat, de dimensiune n.

Vectorul se parcurge secvențial de la primul element, într-o structură DO-WHILE, până când se regăseşte valoarea căutată, sau până la ultimul element, caz în care valoarea căutată nu se află în vector, afişându-se un mesaj corespunzător.

```
Program cautare_prima_aparitie;
Var
  x:array[1..100] of real;  n,i:byte; a:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');   readln(x[i]);
    end;
  Write('Valoarea cautata: ');  readln(a);
  i:=1;
  while (i<=n) and (x(i)<>a) do i:=i+1;
  if i<=n then    writeln('Pozitia = ',i)
          else    writeln('Valoare neregasita !');
End.
```

1.5. Determinarea poziției ultimei apariții a unei valori date într-un vector neordonat, de dimensiune n.

Vectorul se parcurge secvențial într-o structură DO-FOR, de la primul la ultimul element, reținând în aceeaşi variabilă (POZ) valoarea curentă a indicelui, în cazul identității elementului cu valoarea căutată. Dacă variabila POZ are valoarea **0** la sfârşitul ciclării, valoarea căutată nu a fost regăsită printre elementele vectorului şi se afişează un mesaj corespunzător.

```
Program cautare_ultima_aparitie;
Var x:array[1..100] of real; n,i,poz:byte; a:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do begin
            write('X(',i,') = ');
            readln(x[i])
                   end;
  write('Valoarea cautata: ');  readln(a);
  poz:=0;
  for i:=1 to n do  if x[i] = a then poz:=i;
  if poz <> 0 then writeln('Pozitia = ',poz)
      else writeln('Valoare neregasita!')
End.
```

1.6. Determinarea poziției tuturor aparițiilor unei valori date într-un vector neordonat, de dimensiune n.

Vectorul se parcurge secvențial într-o structură DO-FOR, de la primul la ultimul element, reținând valoarea curentă a indicelui în cazul identității elementului cu valoarea căutată, într-un vector (POZ) de poziții (vectorul POZ se construieşte). Dacă la sfârşitul ciclării vectorul POZ este vid (indicele **k** al acestuia este **0**), valoarea căutată nu a fost regăsită şi se afişează un mesaj corespunzător.

```
Program cautare_toate_aparitiile;
Var
  x:array[1..100] of real;  poz:array[1..100] of byte;
  n,i,k:byte; a:real;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = '); readln(x[i]);
    end;
  write('Valoarea cautata: '); readln(a);
  k:=0;
  for i:=1 to n do
    if x[i] = a then
      begin
        k:=k+1;
        poz[k]:=i
      end;
  if k > 0 then
    begin
      write('Pozitiile = ');
      for i:=1 to k do write(poz[i],', ')
    end
           else writeln('Valoare neregasita!');
End.
```

1.7. Căutarea unei valori date într-un şir de numere, ordonat crescător, de dimensiune n.

Pornind de la ipoteza că şirul este ordonat, căutarea valorii se realizează pe subşiruri, în funcție de elementul central al subşirului curent. Rangul elementelor unui subşir poate lua valori în intervalul $[l_s,l_d] \phi [1,n]$, unde l_s este limita stângă şi l_d este limita dreaptă. Dacă valoarea căutată nu coincide cu elementul central al subşirului, procesul de căutare continuă pe subşirul din stânga (dreapta) elementului central, după cum valoarea este mai mică (mai mare) decât acesta, modificându-se corespunzător l_d (l_s) pentru noul subşir.

Modificarea limitei dreapta (stânga) se realizează prin incrementarea (decrementarea) cu o unitate. Rangul elementului central al subşirului curent se determină astfel: $i = \left[\frac{l_s + l_d}{2} \right]$. Procesul de căutare se opreşte când valoarea căutată coincide cu elementul central al subşirului curent, sau când $l_s > l_d$, caz în care valoarea dată nu se regăseşte în şirul inițial.

```
Program cautare_in_vector_ordonat;
Var
  x:array[1..100] of real;
  n,i,s,d:byte;
  a:real;
  vb:boolean;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  write('Valoarea cautata: ');  readln(a);
  s:=1;  d:=n;  vb:=false;
  repeat
    i:=(s+d) div 2;
    if x[i] = a then vb:=true
                else if x[i] < a then s:=i+1
                                 else d:=i-1;
  until (d < s) or vb;
  if vb  then writeln('Pozitia = ',i)
         else writeln('Valoare neregasita!');
End.
```

1.8. Determinarea elementului maxim dintr-un vector de dimensiune n şi a poziției primei (ultimei) sale apariții.

Fie vectorul $X=(x_1,x_2,...,x_n)$. Maximul din vectorul X este $MAX = \max(x_i)$, i=1,n. Deoarece, la un moment dat, comparația se realizează numai pentru două valori, algoritmul poate fi descris astfel:

$MAX_1 = \max(x_1,x_2)$;
$MAX_2 = \max(MAX_1,x_3)$;
$MAX_3 = \max(MAX_2,x_4)$;
...
$MAX_{n-1} = \max(MAX_{n-2},x_n)$;

$MAX = MAX_{n-1}$.

Maximele parţiale MAX_1,MAX_2,...,MAX_{n-1} nu interesează şi de aceea se va utiliza o singură variabilă (MAX) pentru reţinerea elementului maxim din vector. De asemenea, pentru a putea exprima iterativ procesul, variabila MAX va fi iniţializată cu **x(1)**. Algoritmul recursiv poate fi descris astfel:

! formula de start: MAX = x(1);
! formula recursivă: MAX = max(MAX,x(i)), i=2,n.

Pentru a reţine poziţia valorii maxime din vector se utilizează variabila POZ, iniţializată cu **1** (corespunzator poziţiei elementului cu care se iniţializează variabila MAX). În procesul iterativ, variabila POZ se modifică dacă un element x(i) este fie strict mai mare decât MAX (pentru reţinerea poziţiei primei apariţii), fie mai mare sau egal decât MAX (pentru reţinerea poziţiei ultimei apariţii).

```
Program determinare_maxim;
Var
  x:array[1..100] of real;  n,i,poz:byte; max:real;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  max:=x[1]; poz:=1;
  for i:=2 to n do
    if x[i] > max then  begin  max:=x[i];  poz:=i end;
  writeln('Maximul = ',max:10:2,' pe pozitia ',poz)
End.
```

1.9. Determinarea elementului maxim dintr-un vector şi a aparițiilor sale.

Elementul maxim se determină după algoritmul utilizat la exerciţiul 1.8. Deoarece valoarea maximă poate apărea în vector de mai multe ori (cel puţin o dată şi cel mult de **n** ori - în cazul şirului constant), se va construi un vector de poziţii (POZ). Există mai multe variante de rezolvare.

∝ *Varianta 1*. Vectorul POZ se construieşte după ce s-a determinat valoarea maximă, caz în care problema se rezolvă ca în exerciţiul 1.6.

```
Program determinare_pozitii_maxim_1;
Var
  x:array[1..100] of real;
  poz:array[1..100] of byte;
  n,i,k:byte; max:real;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
```

```
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  max:=x[1];
  for i:=2 to n do if x[i] > max then max:=x[i];
  k:=0;
  for i:=1 to n do
    if x[i] = max then
      begin
        k:=k+1;
        poz[k]:=i
      end;
  write('Maximul = ',max:10:2,' pe pozitiile ');
  for i:=1 to k do write(poz[i],'  ')
End.
```

∝ *Varianta 2*. Vectorul POZ se construieşte simultan cu determinarea elementului maxim. Astfel, în urma comparației între variabila MAX şi elementul x(i) apar două cazuri:

- dacă x(i) = MAX, se construieşte un nou element în vectorul de poziții (indicele vectorului POZ se incrementează cu 1);

- dacă x(i) > MAX, înseamnă că s-a găsit o nouă valoare maximă şi indicele vectorului POZ se inițializează cu 1.

Pentru ambele cazuri, în vectorul POZ se reține valoarea curentă a indicelui de parcurgere a vectorului de date.

```
Program determinare_pozitii_maxim_2;
Var
  x:array[1..100] of real; poz:array[1..100] of byte;
  n,i,k:byte; max:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  max:=x[1]; k:=0;
  for i:=1 to n do
    if x[i] >= max then
      begin
        if x[i] > max then begin max:=x[i]; k:=1 end
        else    k:=k+1;
        poz[k]:=i
      end;
```

```
  write('Maximul = ',max:10:2,' pe pozitiile ');
  for i:=1 to k do
    write(poz[i],'  ')
End.
```

1.10. Sortarea unui vector de dimensiune n.

Prin sortare se înțelege aranjarea elementelor unei mulțimi, în ordine crescătoare (descrescătoare) a valorilor. Există mai multe variante de sortare, exemplificate în continuare pe ordonarea crescătoare a elementelor unui vector.

⊄ *Sortarea prin interschimbare.* Se compară două câte două elemente consecutive ale vectorului, interschimbându-le în cazul neîndeplinirii criteriului de ordonare. După o parcurgere integrală a vectorului procesul se reia începând cu primul element. Astfel, elementele cu valoare mică sunt "împinse" către începutul vectorului. De aceea, metoda se mai numeşte "metoda bulelor". Procesul se opreşte când la o parcurgere a vectorului nu s-a produs nici o interschimbare, situație indicată de valoarea de adevăr a unei variabile-semafor (booleană), controlată de programator.

```
Program sortare_prin_interschimbare;
Var
  x:array[1..100] of real;  n,i:byte; aux:real; vb:boolean;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = '); readln(x[i])
    end;
  repeat
    vb:=false;
    for i:=1 to n-1 do
      if x[i] > x[i+1] then
            begin
                  aux:=x[i];
                  x[i]:=x[i+1];
                  x[i+1]:=aux;
                  vb:=true
            end
  until not vb;
  writeln(' Vectorul ordonat este:');
  for i:=1 to n do writeln('X(',i,') = ',x[i]:10:2)
End.
```

⊄ *Sortarea prin selecție.* Metoda presupune determinarea elementului minim din vector şi aducerea lui pe prima poziție, după care se determină minimul din vectorul rămas şi aducerea lui pe a doua poziție etc. Minimul se poate determina

comparând un element al vectorului cu toate care îl succed, interschimbându-le în cazul neîndeplinirii criteriului de ordonare. Această metodă poate avea la rândul ei mai multe variante.

```
Program sortare_prin_selectie;
Var
  x:array[1..100] of real;  n,i,j:byte;   aux:real;
Begin
  write('Dimensiunea vectorului: ');  readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if x[i] > x[j] then
        begin
          aux:=x[i]; x[i]:=x[j];  x[j]:=aux
        end;
  writeln(' Vectorul ordonat este:');
  for i:=1 to n do writeln('X(',i,') = ',x[i]:10:2)
End.
```

∝ *Sortarea prin inserție*. Pornind de la ipoteza că la pasul i elementele predecesoare lui x(i) sunt ordonate, se determină poziția (POZ) în care valoarea lui x(i) se încadrează conform criteriului de ordonare. Elementul x(i) va fi inserat în acea poziție, după ce toate elementele vectorului, începând cu poziția POZ şi până la sfârşit, glisează cu o poziție la dreapta. Se reduce astfel numărul de interschimbări, deoarece, în cazul în care un element x(i) este mai mare decât precedentul, acesta se consideră ordonat.

```
Program sortare_prin_insertie;
Var
  x:array[1..100] of real;  n,i,j,k,poz:byte;   aux:real;
Begin
  write('Dimensiunea vectorului: '); readln(n);
  writeln('Elementele vectorului: ');
  for i:=1 to n do
    begin
      write('X(',i,') = ');
      readln(x[i])
    end;
  for i:=2 to n do
    if x[i] < x[i-1] then
      begin
        poz:=0; j:=1;
```

```
        repeat  if x[i]<x[j] then poz:=j  else j:=j+1
        until (j>n) or (poz<>0);
        aux:=x[i];
        for k:=i downto poz+1 do  x[k]:=x[k-1];
        x[poz]:=aux
      end;
  writeln(' Vectorul ordonat este:');
  for i:=1 to n do  writeln('X(',i,') = ',x[i]:10:2)
End.
```

1.11. Interclasarea a doi vectori de dimensiuni variabile.

Prin interclasare se înțelege procesul de obținere din două sau mai multe mulțimi ordonate o nouă mulțime, ordonată după acelaşi criteriu. Există mai multe variante de interclasare, exemplificate în continuare pe doi vectori, sortați crescător.

⊄ *Varianta 1* presupune compararea a două elemente, câte unul din fiecare vector inițial, cu scrierea celui mai mic dintre ele în vectorul rezultant şi trecerea la următorul element al vectorului inițial din care s-a preluat. Procesul de comparare se încheie când s-a epuizat unul din vectorii inițiali. În continuare, elementele netransferate ale celuilalt vector inițial se copiază în vectorul rezultant.

```
Program interclasare_vectori_metoda_1;
Var
  x:array[1..100] of real;
  y:array[1..150] of real;
  z:array[1..250] of real;
  m,n,i,j,k,l:byte;
Begin
  write('Dimensiunea vectorului 1: '); readln(m);
  writeln('Elementele vectorului 1: ');
  for i:=1 to m do
       begin write('X(',i,') = '); readln(x[i]) end;
  write('Dimensiunea vectorului 2: '); readln(n);
  writeln('Elementele vectorului 2: ');
  for i:=1 to n do
       begin write('Y(',i,') = '); readln(y[i]) end;
  k:=0; i:=1; j:=1;
  while (i <= m) and (j <= n) do
    begin
      k:=k+1;
      if x[i] < y[j] then
        begin z[k]:=x[i]; i:=i+1 end
                    else
        begin z[k]:=y[j]; j:=j+1 end
    end;
  if i>m then
    for l:=j to n do
```

```
      begin
        k:=k+1; z[k]:=y[l]
      end
          else
    for l:=i to m do
      begin
        k:=k+1; z[k]:=x[l]
      end;
  writeln(' Vectorul interclasat este:');
  for i:=1 to m+n do
    writeln('Z(',i,') = ',z[i]:10:2)
End.
```

∝ *Varianta 2* presupune obținerea vectorului rezultant într-un proces unic de comparare. Pentru a continua procesul în cazul în care se epuizează unul din vectorii inițiali, ultimul element al acestuia va primi o valoare mai mare decât oricare din valorile regăsite în vectorii inițiali. Această valoare poartă denumirea de *high-value* (HV) şi depinde de natura şi reprezentarea internă a vectorilor. Comparările ulterioare cu HV vor forța copierea în vectorul rezultant numai a elementelor vectorului care nu s-a epuizat. Procesul se încheie când ambii vectori inițiali au fost parcurşi integral, deci elementele finale au valoarea HV.

```
Program interclasare_vectori_metoda_2;
Var
  x:array[1..101] of real;
  y:array[1..151] of real;
  z:array[1..250] of real;
  m,n,i,j,k:byte;
Const
  hv=1E10;
Begin
  write('Dimensiunea vectorului 1: '); readln(m);
  writeln('Elementele vectorului 1: ');
  for i:=1 to m do begin
                        write('X(',i,') = '); readln(x[i])
                    end;
  write('Dimensiunea vectorului 2: '); readln(n);
  writeln('Elementele vectorului 2: ');
  for i:=1 to n do begin
                        write('Y(',i,') = '); readln(y[i])
                    end;
  k:=0; i:=1; j:=1;
  while (x[i]<>hv) or (y[j]<>hv) do
    begin
      k:=k+1;
      if x[i] < y[j] then begin
                              z[k]:=x[i];
                               i:=i+1;
```

```
                              if i>m then x[i]:=hv
                            end
      else begin
              z[k]:=y[j]; j:=j+1; if j>n then y[j]:=hv
           end
    end;
  writeln(' Vectorul interclasat este:');
  for i:=1 to m+n do writeln('Z(',i,') = ',z[i]:10:2)
End.
```

Observație: Pentru a nu altera valoarea elementului final al fiecărui vector inițial se va rezerva o poziție suplimentară la declararea masivelor.

1.12. Suma elementelor unei matrice dreptunghiulare de dimensiuni mHn.

Fie matricea A = $\begin{pmatrix} a_{11}\, a_{12} \cdots a_{1n} \\ a_{21}\, a_{22} \cdots a_{2n} \\ \cdots\cdots\cdots\cdots \\ a_{m1}\, a_{m2} \cdots a_{mn} \end{pmatrix}$. Suma elementelor este:

$S=a_{11}+a_{12}+...+a_{1n}+a_{21}+...+a_{2n}+...+a_{m1}+...+a_{mn}=\sum_{i=1}^{m}\sum_{j=1}^{n} a(i,j)$, dacă se însumează pe linii (lexicografic) sau

$S=a_{11}+a_{21}+...+a_{m1}+a_{12}+...+a_{m2}+...+a_{1n}+...+a_{mn}=\sum_{j=1}^{n}\sum_{i=1}^{m} a(i,j)$, dacă se însumează pe coloane (invers lexicografic). Matricea A poate fi privită ca un vector de dimensiune m·n dacă este liniarizată (lexicografic sau invers lexicografic). Pornind de la ipotezele formulate la determinarea sumei elementelor unui vector, algoritmul (parcurgând matricea lexicografic), poate fi descris astfel:

```
        S = 0
    -------------------------
        j=1     S = S + a11 = a11
        j=2     S = S + a12 = a11 + a12
i=1     ...
        j=n     S = S + a1n = a11 +...+ a1n
    -------------------------
        j=1     S = S + a21 = a11 +...+ a21
i=2     ...
        j=n     S = S + a2n = a11 +...+ a2n
    ……….
        j=1     S = S + am1 = a11 +...+ am1
i=m     ...
        j=n     S = S + amn = a11 +...+ amn
    -------------------------
```

Algoritmul recursiv poate fi descris astfel:

- formula de start: S = 0;
- formula recursivă: S = S + a(i,j); i=1,m; j=1,n

Parcurgând matricea invers lexicografic algoritmul este similar, cu deosebirea că indicele liniilor (**i**) va lua toate valorile din intervalul [1,m] pentru o valoare dată a indicelui de coloane (**j**). Elementele matricei se introduc de la tastatură, element cu element, cu ajutorul a două structuri DO-FOR imbricate.

```
Program suma_elemente_matrice;
Var  a:array[1..10,1..20] of real;  m,n,i,j:byte; s:real;
Begin
  write('Numarul de linii: '); readln(m);
  write('Numarul de coloane: '); readln(n);
  writeln('Elementele matricei: ');
  for i:=1 to m do
    for j:=1 to n do
      begin
        write('A(',i,',',j,') = ');
        readln(a[i,j]);
      end;
  s:=0;
  for i:=1 to m do
    for j:=1 to n do
      s:=s+a[i,j];
  writeln(' Suma: ',s:15:3)
End.
```

1.13. Determinarea elementelor maxim şi minim dintr-o matrice dreptunghiulară, de dimensiuni mHn.

Fie matricea A=$\begin{pmatrix} a_{11}\,a_{12}\cdots a_{1n} \\ a_{21}\,a_{22}\cdots a_{2n} \\ \cdots\cdots\cdots\cdots\cdots \\ a_{m1}\,a_{m2}\cdots a_{mn} \end{pmatrix}$. Liniarizând matricea lexicografic se obține un vector de dimensiune mHn: A=(a_{11},a_{12},...,a_{1n},a_{21},...,a_{2n},...,a_{m1},...,a_{mn}). Elementele maxim şi minim se determină astfel: MAX=max(a_{11},a_{12},...,a_{1n},a_{21},...,a_{2n},...,a_{m1},...,a_{mn}) şi MIN= min(a_{11},a_{12},...,a_{1n},a_{21},...,a_{2n},...,a_{m1},...,a_{mn}). Algoritmul recursiv poate fi descris astfel:

- formulele de start: MAX = a(1,1); MIN = a(1,1);
- formulele recursive: MAX = max{MAX, a(i,j)}; i=1,m; j=1,n;
  MIN = min{MIN, a(i,j)}; i=1,m; j=1,n.

Dacă matricea se consideră liniarizată invers lexicografic algoritmul este similar, cu deosebirea că ordinea de variație a indicilor este inversă. Maximul şi minimul se vor determina concomitent, pornind de la ipoteza că un element oarecare a(i,j) se poate afla:

- în intervalul (MAX, +4) şi este noua valoare maximă;
- în intervalul (-4, MIN) şi este noua valoare minimă;
- în intervalul [MIN, MAX], caz în care nu afectează nici una din valorile căutate.

```
Program minim_maxim_din_matrice;
Var
  a:array[1..10,1..20] of real;
  m,n,i,j:byte;
  min,max:real;
Begin
  write('Numarul de linii: ');
  readln(m);
  write('Numarul de coloane: ');
  readln(n);
  writeln('Elementele matricei: ');
  for i:=1 to m do
    begin
      write('Linia ',i,': ');
      for j:=1 to n do
        read(a[i,j])
    end;
  max:=a[1,1];
  min:=max;
  for i:=1 to m do
    for j:=1 to n do
      if a[i,j] > max then max:=a[i,j]
      else if a[i,j] < min then min:=a[i,j];
  writeln(' Minim = ',min:15:3);
  writeln(' Maxim = ',max:15:3)
End.
```

1.14. Determinarea sumei elementelor de pe fiecare coloană a unei matrice dreptunghiulare precum şi a valorii maxime dintre aceste sume.

Fie matricea A = $\begin{pmatrix} a_{11}\, a_{12} \cdots a_{1n} \\ a_{21}\, a_{22} \cdots a_{2n} \\ \cdots\cdots\cdots\cdots \\ a_{m1}\, a_{m2} \cdots a_{mn} \end{pmatrix}$. Însumând elementele de pe fiecare coloană a

$$\overset{\$}{S} = (\overset{\$}{S}_1,\ \overset{\$}{S}_2,\ \ldots,\ \overset{\$}{S}_n)$$

unei matrice se obține un vector (S) de sume, de dimensiune n. Fiecare element al acestui vector se determină după algoritmul descris la exercițiul 1, asfel:

! formula de start: $S(j) = 0$;

! formula recursivă: $S(j) = S(j) + a(i,j)$; $i=1,m$.

Determinarea sumei maxime se poate realiza ❶ în etape distincte, prin construirea vectorului S urmată de determinarea maximului dintr-un vector sau ❷ în aceeaşi

structură repetitivă astfel: se determină suma elementelor de pe o coloană a matricei, după care, dacă este prima coloană se aplică formula de start MAX=S(1), altfel, formula recursivă MAX=max{MAX,S(j)}, j=2,n.

```
Program sume_pe_coloane;
Var
  a:array[1..10,1..20] of real;  s:array[1..20] of real;
  m,n,i,j:byte; max:real;
Begin
  write('Numarul de linii: '); readln(m);
  write('Numarul de coloane: '); readln(n);
  writeln('Elementele matricei: ');
  for i:=1 to m do
    for j:=1 to n do
      begin  write('A(',i,',',j,') = '); readln(a[i,j]) end;
  for j:=1 to n do
    begin     s[j]:=0;
              for i:=1 to m do s[j]:=s[j]+a[i,j];
              if j = 1 then  max:=s[j]
              else if s[j] > max then  max:=s[j]
    end;
  writeln('Sumele pe coloane sunt:');
  for j:=1 to n do writeln(' Coloana ',j,' = ',s[j]:15:3);
  writeln(' Maximul   = ',max:15:3)
End.
```

1.15. Determinarea elementului maxim de pe fiecare linie şi a elementului maxim dintr-o matrice dreptunghiulară de dimensiuni mHn.

Fie matricea $A=\left(\begin{array}{c} a_{11}\, a_{12} \cdots\ a_{1n} \\ a_{21}\, a_{22} \cdots\ a_{2n} \\ \cdots\cdots\cdots\cdots\cdots \\ a_{m1}\, a_{m2} \cdots\ a_{mn} \end{array}\right)\begin{array}{l} \rightarrow \mathrm{MAX}_1 \\ \rightarrow \mathrm{MAX}_2 \\ \cdots\cdots \\ \rightarrow \mathrm{MAX}_m \end{array}$. Elementele maxime de pe fiecare linie vor forma un vector de dimensiune m: MAX = (MAX_1,MAX_2,...,MAX_m). Un element al vectorului MAX se determină astfel:

- formula de start: MAX(i) = a(i,1);
- formula recursivă: MAX(i) = max{MAX(i),a(i,j)}, j=2,n.

Maximul din matrice (MAXM) se poate determina ❶ în etape distincte, prin determinarea elementului maxim al vectorului MAX după ce acesta a fost construit: MAXM=max{MAX(i)}, i=1,m sau ❷ în aceeaşi structură repetitivă, comparând elementul MAX(i) recent determinat, cu variabila MAXM.

```
Program maxime_pe_linii;
Var
  a:array[1..10,1..20] of real; max:array[1..10] of real;
  m,n,i,j:byte; maxm:real;
```

```
Begin
  write('Numarul de linii: ');  readln(m);
  write('Numarul de coloane: ');  readln(n);
  writeln('Elementele matricei: ');
  for i:=1 to m do
    for j:=1 to n do
      begin
        write('A(',i,',',j,') = '); readln(a[i,j]);
      end;
   for i:=1 to m do
     begin
       max[i]:=a[i,1];
       for j:=2 to n do
           if max[i] < a[i,j] then
               max[i]:=a[i,j];
       if (i=1) or (maxm < max[i]) then
               maxm:=max[i]; {formula de start sau recursiva}
      end;
  writeln(' Maximele pe linii sunt:');
  for i:=1 to m do writeln('Linia ',i,' = ',max[i]:10:2);
  writeln('Maximul din matrice = ',maxm:10:2)
End.
```

1.16. Numărarea liniilor unei matrice dreptunghiulare de dimensiuni mHn ale căror elemente sunt în ordine strict descrescătoare.

Matricea $\mathbf{A_{mHn}}$ va fi parcursă lexicografic. Pentru fiecare linie se verifică, într-o structură repetitivă WHILE-DO, dacă oricare două elemente succesive satisfac criteriul cerut. La prima neconcordanţă se iese forţat din ciclare. Numărătorul de linii care au proprietatea cerută (NR) se incrementează cu **1** numai dacă s-a parcurs întreaga linie (j>n-1). Altfel, linia nu satisface cerinţa şi procesul se reia pentru următoarea linie a matricei. În final, dacă variabila NR are valoare nenulă va fi afişată pe ecran, altfel se afişează un mesaj corespunzător.

```
Program numarare_linii;
Var
  a:array[1..10,1..20] of real;  m,n,i,j,nr:byte;
Begin
  write('Numarul de linii: '); readln(m);
  write('Numarul de coloane: '); readln(n);
  writeln('Elementele matricei: ');
  for i:=1 to m do
    for j:=1 to n do
      begin
        write('A(',i,',',j,') = ');
        readln(a[i,j])
      end;
  nr:=0;
```

```
  for i:=1 to m do
    begin
      j:=1;
      while (j<=n-1) and (a[i,j]>a[i,j+1]) do j:=j+1;
      if j > n-1 then   nr:=nr+1;
    end;
  if nr > 0 then
      writeln('Nr. de linii: ',nr)
            else
      writeln('Nu exista linii cu elemente descrescatoare !');
End.
```

1.17. Determinarea poziției primei apariții a unei valori date într-o matrice dreptunghiulară, de dimensiuni mHn.

Matricea va fi parcursă în ordinea lexicografic. În momentul în care se găseşte valoarea căutată se afişează poziția (linia şi coloana elementului respectiv) şi se abandonează parcurgerea matricei. Ieşirea "forțată" din structurile repetitive se realizează cu ajutorul unei variabile booleene (VB), care ia valoarea TRUE dacă valoarea dată a fost regăsită printre elementele matricei sau FALSE în caz contrar. Testarea suplimentară a variabilei VB, alături de condiția de sfârşit de linii (i>m) şi sfârşit de coloane (j>n), transformă structurile repetitive cu numărător (gândite pentru parcurgerea matricei) în două structuri WHILE-DO imbricate. În final, dacă VB este falsă, înseamnă că valoarea nu a fost regăsită şi se afişează un mesaj corespunzător.

```
Program pozitia_primei_aparitii;
Var
  a:array[1..10,1..20] of real;
  m,n,i,j:byte;   x:real;  vb:boolean;
Begin
  Write('Numarul de linii: ');   Readln(m);
  Write('Numarul de coloane: '); Readln(n);
  Writeln('Elementele matricei: ');
  For i:=1 to m do
    For j:=1 to n do
      begin
        write('A(',i,',',j,') = ');
        readln(a[i,j])
      end;
  Write('Valoarea cautata: '); readln(x);
  vb:=false; i:=1;
  While (i<=m) and not vb do
    begin
      j:=1;
      while (j<=n) and not vb do
        if a[i,j] = x then
          begin
           vb:=true;
```

```
           writeln('Valoarea apare pe linia ',i,' coloana ',j)
          end
        else  j:=j+1;
      i:=i+1
    end;
  If not vb then writeln('NU exista valoarea cautata!')
End.
```

1.18. Înmulțirea a două matrice dreptunghiulare cu elemente reale de dimensiuni mHn, respectiv nHp.

Fie matricele A={a(i,k)}, i=1,m, k=1,n şi B={b(k,j)}, k=1,n, j=1,p. Matricea rezultat este C=AHB={c(i,j)}, i=1,m, j=1,p. ∨inând cont că înmulțirea de matrice nu este comutativă şi că elementul neutru pentru adunarea în numere reale este **0**, algoritmul pentru determinarea unui element c(i,j) poate fi descris astfel:

- formula de start: c(i,j) = 0;
- formula recursivă: c(i,j) = c(i,j) + a(i,k)Hb(k,j); k=1,n.

Se obține o construcție compusă din trei structuri repetitive cu numărător, imbricate.

```
Program inmultire_matrice;
Var
  a:array[1..10,1..20] of real;
  b:array[1..20,1..30] of real;
  c:array[1..10,1..30] of real;
  m,n,p,i,j,k:byte; x:real;
  vb:boolean;
 Begin
 Write('Nr. de linii ale matricei deinmultit: '); Readln(m);
 Write('Nr. de coloane ale matricei deinmultit: '); Readln(n);
 Write('Nr. de coloane ale matricei inmultitor: '); Readln(p);
 Writeln('Elementele matricei deinmultit: ');
 For i:=1 to m do
    For j:=1 to n do
      begin
        write('A(',i,',',j,') = ');  readln(a[i,j])
      end;
 Writeln('Elementele matricei inmultitor: ');
 For i:=1 to n do
    For j:=1 to p do
      begin
        write('B(',i,',',j,') = ')   readln(b[i,j])
      end;
 For i:=1 to m do
    For j:=1 to p do
      begin
        c[i,j]:=0;
        for k:=1 to n do c[i,j]:=c[i,j]+a[i,k]*b[k,j]
      end;
```

```
 Writeln('Matricea rezultat este:');
 For i:=1 to m do
    begin
      for j:=1 to p do  write(c[i,j]:7:2,' ');
      writeln
    end
End.
```

Observație: Matricele A şi B se introduc câte un element pe un rând al ecranului, iar matricea C se va afişa câte o linie pe un rând al ecranului.

1.19. Rezolvarea ecuației $ax^2 + bx + c = 0$, a,b,c 0 R.

Având în vedere că de la tastatură se poate introduce orice triplet (a,b,c) de valori reale, rezolvarea ecuației date necesită următoarea discuție, după valorile coeficienților:

a. (0,0,0) - ecuația este nedeterminată;

b. (0,0,c□0) - ecuația este imposibilă;

c. (0,b□0,c0R) - ecuația este degenerată şi se determină soluția ecuației de gradul I;

d. (a□0,b0R,c0R) - ecuația este determinată, de gradul al II-lea, iar discuția se va face după semnul discriminantului $d=b^2-4ac$, astfel:

d.1. dacă d>0, ecuația are două soluții reale distincte;

d.2. dacă d=0, ecuația are soluție dublă;

d.3. dacă d<0, ecuația are soluții imaginare.

Soluțiile vor fi determinate şi afişate pe ecran.

```
Program ecuatie;
Var  a,b,c,x,x1,x2,r,im,d:real;
Begin
  Write('Coeficientii ecuatiei: '); Readln(a,b,c);
  If a<>0 then
    begin
      d:=b*b-4*a*c;
      if d>=0 then
        begin
          if d>0 then
            begin
              x1:=(-b+sqrt(d))/2/a; x2:=(-b-sqrt(d))/2/a;
              writeln('X1=',x1:6:2,'   X2=',x2:6:2)
            end
          else
            begin
              x:=-b/2/a; writeln('X1 = X2 = ',x:6:2)
            end;
        end
      else
```

```
        begin
          r:=-b/2/a; im:=abs(sqrt(-d)/2/a);
          write('X1=',r:6:2,'+',im:6:2,'i; X2=',
                 r:6:2,' -',im:6:2,'i')
        end;
    end
  else if b<>0 then
    begin
      x:=-c/b;
      writeln('Ecuatie de gradul I:  X=',x:6:2)
    end
  else if c<>0 then writeln('Ecuatie imposibila !')
  else writeln('Ecuatie nedeterminata !')
End.
```

1.20. Determinarea c.m.m.d.c. dintre două numere naturale nenule.

Pentru determinarea c.m.m.d.c. dintre două numere (A şi B) se aplică algoritmul lui Euclid, care constă în următoarele etape:

a. primul deîmpărțit (D) este A, iar primul împărțitor (I) este B;

b. se împarte D la I şi se obține un cât (C) şi un rest (R), conform teoremei împărțirii: D=IHC+R;

c. dacă R=0, atunci c.m.m.d.c. este ultimul împărțitor (I) şi procesul se opreşte.

d. dacă R=1, atunci numerele sunt prime între ele şi procesul se opreşte;

e. dacă R∃2, noul deîmpărțit este vechiul împărțitor, noul împărțitor este restul, iar procesul se reia de la pasul b.

În program, câtul împărțirii nu se calculează, având în vedere existența în Pascal a operatorului MOD, care returnează restul împărțirii întregi a două numere naturale.

```
Program cmmdc_al_doua_numere;
Var
  a,b,d,i,r:word;
Begin
  write('Numerele: '); readln(a,b);
  d:=a; i:=b;
  repeat
    r:=d mod i;
    if r=0 then  writeln('C.m.m.d.c. = ',i)
    else if r=1 then writeln('Numere prime intre ele !')
    else begin d:=i; i:=r end
  until r < 2
End.
```

1.21. Determinarea c.m.m.d.c. dintr-un şir de numere naturale nenule.

Fie şirul $\mathbf{X=(x_1,x_2,...,x_n)}$. Pentru determinarea c.m.m.d.c. al şirului X se foloseşte proprietatea acestuia de asociativitate, astfel:

$CMMDC_1$ = cmmdc(x_1, x_2);
$CMMDC_2$ = cmmdc($CMMDC_1$, x_3);
...
$CMMDC_{n-1}$= cmmdc($CMMDC_{n-2}$, x_n);

CMMDC = $CMMDC_{n-1}$,

unde c.m.m.d.c. pentru două numere se determină conform exercițiului 1.20. Pentru că cei mai mari divizori comuni parțiali $CMMDC_1$, $CMMDC_2$,..., $CMMDC_{n-1}$ sunt de fapt împărțitorii curenți, nu se folosesc variabile suplimentare. Procesul se opreşte fie când şirul de numere a fost epuizat, caz în care c.m.m.d.c. a fost determinat, fie când în urma unei împărțiri restul este **1**, caz în care numerele sunt prime între ele.

```
Program cmmdc_al_unui_sir_de_numere;
Var
  x:array[1..20] of word;  k,n,d,i,r:word;
Begin
  write('Dimensiunea sirului: '); readln(n);
  write('Numerele: ');
  for i:=1 to n do read(x[i]);
  d:=x[1]; r:=0; k:=2;
  while (r<>1) and (k<=n) do
    begin
      i:=x[k];
      repeat
          r:=d mod i;   d:=i;   i:=r
      until r < 2;
      k:=k+1
    end;
  if r<>1 then writeln('C.m.m.d.c. = ',d)
  else writeln('Numere prime intre ele !')
End.
```

1.22. Fie o matrice A_{mxn} care reprezintă notele obținute de m studenți la n discipline. Să se determine:

a) studenții integralişti (care au toate notele ≥ 5);
b) studenții bursieri (integraliştii cu media ≥ 8,50);
c) disciplinele la care s-au înregistrat cei mai mulți restanțieri;
d) media pe fiecare disciplină (se iau în calcul doar notele de promovare).

```
program note;
uses crt;
var a:array[1..15,1..20] of 0..10;
    med:array[1..20] of real;
    dsp:array[1..20] of 1..20;
    stb:array[1..15] of real;
    sti:array[1..15] of 1..15;
    m,n,i,j,k,maxr,nr:byte;
```

```
    vb:boolean;
begin
     clrscr;
     write('Nr. de studenti:');readln(m);
     write('Nr. de discipline:');readln(n);
     for i:=1 to m do
      for j:=1 to n do begin
            write('Nota studentului ',i,' la disciplina ',
                   j,' :');
            readln(a[i,j])
                       end;
     k:=0;
     for i:=1 to m do begin
         j:=1;
         while (j<=n) and (a[i,j]>=5) do inc(j);
         if j>n then
                  begin
                        k:=k+1;
                        sti[k]:=i
                  end
                      end;
     if k=0 then writeln('Nu exista studenti integralisti!')
     else begin
               writeln('Studentii integralisti sunt:');
               for i:=1 to k do write(sti[i]:4)
          end;
     writeln('STUDENTII BURSIERI');
     writeln;
     if k=0 then writeln('Nu exista bursieri!')
     else
       begin
               vb:=false;
               for i:=1 to k do begin
                   stb[i]:=0;
                   for j:=1 to n do
                                stb[i]:=stb[i]+a[sti[i],j];
                   stb[i]:=stb[i]/n;
                   if stb[i]>=8.5 then
                      begin
                        vb:=true;
                        writeln('Studentul ',sti[i],
                        ' este bursier cu media',stb[i]:6:2);
                        readln
                      end
                                end;
               if not vb then writeln('Nu exista bursieri!');
               readln
          end;
     writeln;
     writeln('DISCIPLINELE CU CEI MAI MULTI RESTANTIERI');
```

```
    writeln;
    for j:=1 to n do begin
        k:=0;
        for i:=1 to m do if a[i,j]<5 then inc(k);
        if j=1 then begin
                        maxr:=k;
                        nr:=1;
                        dsp[nr]:=j
                    end
        else if k>maxr then begin
                                maxr:=k;
                                nr:=1;
                                dsp[nr]:=j
                            end
             else if k=maxr then begin
                     nr:=nr+1;
                     dsp[nr]:=j
                                end;
                     end;
    writeln('Disciplinele sunt:');
    for i:=1 to nr do write(dsp[i]:4);
    readln;
    writeln;
    writeln('MEDIA PE FIECARE DISCIPLINA');
    writeln;
    for j:=1 to n do begin
        med[j]:=0;
        k:=0;
        for i:=1 to m do
            if a[i,j]>=5 then
                            begin
                                 inc(k);
                                 med[j]:=med[j]+a[i,j]
                             end;
        if k>0 then med[j]:=med[j]/k
                    end;
    for j:=1 to n do
        if med[j]<>0 then
                  writeln('La disciplina ',j,
                          ' - media a fost',med[j]:6:2)
        else
                  writeln('La disciplina ',j,
                          ' nu a promovat nici-un student');
    readln
end.
```