DIACONU DIANA-ELENA

ISTRATE NICOLAE-CECILIAN (in memoriam)

CHIRIAC BEATRICE-MIHAELA

PROGRAMAREA CALCULATOARELOR ÎN LIMBAJUL C++

Auxiliar curricular pentru disciplina Informatică

Clasa a X-a

Acest material este distribuit gratuit în format electronic

Decembrie 2024
Târgoviște

CUPRINS

Capitolul 1

Tablouri unidimensionale (vectori)

Noțiuni teoretice

Un tablou unidimensional (vector) este o structură de elemente de acelaşi tip plasate în locaţii de memorie alăturate şi care pot fi accesate cu ajutorul unor indici.

În memoria calculatorului o variabilă de tip întreg ocupă doi octeţi. Un vector care conţine 4 elemente de tip întreg are 2*4 octeţi alăturaţi. Pentru a putea accesa elementele unui vector folosim un singur identificator cu indici diferiţi.

Un vector se defineşte prin numărul de linii n şi elementele v[i] unde $i \in \{1, 2, ..n\}$

De exemplu, mai jos avem un vector, pe care îl notăm cu v, care are 4 elemente întregi:

5	-2	27	63	v
v[0]	v[1]	v[2]	v[3]	

Acest vector se declară int v[4], iar elementele lui sunt v[0], v[1], v[2], v[3], unde v[0]=5, v[1]= -2, v[2]=27, v[3]=63.

Probleme rezolvate

Problema 1. Scrieţi un program în C++ în care se citeşte n număr întreg şi n elemente întregi ale unui vector. Să se afişeze elementele vectorului în ordinea în care au fost introduse.

#include <iostream>
using namespace std;
int main()
{int v[10], i, n;
cout<<"n="; cin>>n;
for(i=0;i<=n-1;i++)
{
cout<<"v["<<i<<"]=";
	se citeşte n, numărul de elemente al vectorului
	se citesc elementele vectorului

cin>>v[i];
}
for(i=0;i<=n-1;i++)
cout<<v[i]<<", ";
return 0;
}
	se afişează elementele vectorului cu virgulă între ele

Problema 2. Scrieţi un program în C++ în care se citeşte n număr întreg şi n elemente întregi ale unui vector şi se afişează elementele vectorului în ordinea inversă introducerii lor.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n;
 cout<<"n="; cin>>n;
  for(i=0;i<=n-1;i++)
     cin>>v[i];
  for(i=n-1;i>=0;i--)
     cout<<v[i]<<", ";
          return 0;}
```

Problema 3. Scrieţi un program în C++ în care se citeşte n număr întreg şi n elemente întregi ale unui vector şi se afişează suma elementelor vectorului.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, s;
 cout<<"n="; cin>>n;
 s=0;
  for(i=0;i<=n-1;i++)
     {cin>>v[i]; s=s+v[i];}
 cout<<"s="<<s;
 return 0;}
```

Problema 4. Scrieţi un program în C++ în care se citeşte n număr întreg şi n elemente întregi ale unui vector şi se afişează suma elementelor pare de pe poziţii impare din vector.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, s;
 cout<<"n="; cin>>n;
 s=0;
```

```
 for(i=0;i<=n-1;i++)
    {cin>>v[i];
     if(v[i]%2==0 && i%2==1)
       s=s+v[i];}
 cout<<"Numerotarea incepe de la zero"<<'\n';
 cout<<"s="<<s;
 return 0;}
```

Problema 5. Scrieţi un program în C++ în care se citeşte n număr întreg şi n elemente întregi ale unui vector şi se afişează media aritmetică a elementelor impare dintr-un vector.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, s, nr;
 cout<<"n="; cin>>n;
 s=0; nr=0;
  for(i=0;i<=n-1;i++)
     {cin>>v[i];
       if(v[i]%2!=0)
         {s=s+v[i]; nr=nr+1;}
      }
 cout<<"media="<<(float)s/nr;
 return 0;
}
```

Problema 6. Scrieţi un program în C++ în care se citeşte un număr nr şi un vector cu n elemente întregi. Să se afişeze poziţia din vector pe care se află, iar dacă nu, să se afişeze un mesaj.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, nr, ok=0;
 cout<<"n="; cin>>n;
 cout<<"nr="; cin>>nr;
 for(i=0; i<=n-1 && ok==0; i++)
  {cin>>v[i];
   if(v[i]==nr)
    {ok=1;
      cout<<nr<<" este pe pozitia "<<i+1;}
  }
  if(ok==0)
     cout<<nr<<" nu este in vector";
 return 0;}
```

Observație: Căutarea în vector s-a făcut de la stânga la dreapta şi s-a afişat prima poziţie pe care s-a găsit nr în vector.

Problema 7. Scrieţi un program în C++ în care se citeşte un număr nr şi un vector cu n elemente întregi. Să se afişeze minimul din elementele întregi ale unui vector.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, min;
 cout<<"n="; cin>>n;
 for(i=0;i<=n-1;i++)
   cin>>v[i];
 min=v[0];
 for(i=1;i<=n-1;i++)
   if(v[i]<min)
      min=v[i];
 cout<<"min="<<min;
 return 0;}
```

Problema 8. Se citeşte un vector cu n elemente întregi şi un număr nr. Să se insereze nr pe prima poziţie în vector.

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, n, nr;
 cout<<"n="; cin>>n;
 cout<<"nr="; cin>>nr;
 for(i=0;i<=n-1;i++)
  cin>>v[i];
n=n+1;
 for(i=n-1;i>=1;i--)
   v[i]=v[i-1];
 v[0]=nr;
  for(i=0;i<=n-1;i++)
   cout<<v[i]<<", ";
 return 0;}
```

Observație: Am crescut numărul de elemente ale vectorului cu unu, după care am deplasat toate elementele spre dreapta cu o unitate. Ultimul element introdus în vector a fost nr pe care l-am adăugat pe prima poziţie.

Problema 9. Se citeşte un vector cu n elemente întregi şi un număr nr. Să se copieze elementele unui vector în alt vector.

```
#include <iostream>
using namespace std;
int main()
{int A[10], B[10],i, j, n;
 cout<<"n="; cin>>n;
 for(i=0;i<=n-1;i++)
  cin>>A[i];
for(i=0;i<=n-1;i++)
  B[i]=A[i];
for(i=0;i<=n-1;i++)
   cout<<B[i]<<", ";
return 0;}
```

Observație: Atribuirea elementelor unui vector altui vector nu se face B=A ci element cu element.

Problema 10. Se citeşte un vector cu n elemente întregi. Să se afişeze vectorul permutat circular spre stânga.

Dacă iniţial vectorul a fost 1, 2, 3, se va afişa:
2, 3, 1,
3, 1, 2,
1, 2, 3,

```
#include <iostream>
using namespace std;
int main() //permutari circulare
{int v[100], i, j, t, n, aux;
 cout<<"n="; cin>>n;
 for(i=0;i<=n-1;i++)
     cin>>v[i];
  for(i=0;i<=n-1;i++)
  { aux=v[0];
    for(j=1;j<=n-1;j++)
       v[j-1]=v[j];
    v[n-1]=aux;
   for(t=0;t<=n-1;t++)
     cout<<v[t]<<", ";
     cout<<'\n';
  }
 return 0;}
```

Problema 11. Ordonarea elementelor unui vector

De multe ori avem nevoie ca elementele unui vector să fie ordonate crescător sau descrescător. Pentru ordonarea vectorilor sunt mai multe metode: Bubble sort (metoda bulelor sau interschimbare), minim (maxim), inserţie, interclasare, Quicksort (sortare rapidă).

Metoda Bubble sort:

```
#include <iostream>
using namespace std;
int main()
{int v[10], n, i, ok, aux;
 cout<<"n="; cin>>n;
 for(i=0;i<=n-1;i++)
  cin>>v[i];
do{    ok=1;
       for(i=0;i<=n-2;i++)
         if(v[i]>v[i+1])
           {aux=v[i];
            v[i]=v[i+1];
            v[i+1]=aux;
            ok=0;}
  }while(ok= =0);
 for(i=0;i<=n-1;i++)
   cout<<v[i]<<", ";
 return 0;}
```

Rulare
Dacă iniţial datele citite au fost: n=1,
v[4] este 9 -2 4 1
Paşii rulaţi sunt:
ok=1
i=0, v[0]>v[1], ok=0, vectorul va fi -2 9 4 1

i=1, v[1]>v[2], ok=0, vectorul va fi -2 4 9 1

i=2, v[2]>v[3], ok=0, vectorul va fi -2 4 1 9

ok=0 este adevărat, se repetă bucla do while
i=0, v[0]>v[1] fals
i=1, v[1]>v[2], ok=0, vectorul va fi -2 1 4 9
i=2, v[2]>v[3] fals

ok=0 este adevărat, se repetă bucla do while
i=0, v[0]>v[1] fals
i=1, v[1]>v[2] fals
i=2, v[2]>v[3] fals

ok=0 este fals se termină bucla repetitivă şi se afişează vectorul -2 1 4 9

Metoda minimului:

```
#include <iostream>
using namespace std;
int main()
{int v[10], i, j, n, min, poz, aux;
 cout<<"n="; cin>>n;
 for(i=0;i<=n-1;i++)
     cin>>v[i];
 for(i=0;i<=n-2;i++)
 { min=v[i]; poz=i;
   for(j=i+1;j<=n-1;j++)
     if(v[j]<min)
       { min=v[j];
         poz=j;
       }
   aux=v[i];
   v[i]=v[poz];
   v[poz]=aux;
 }
 for(i=0;i<=n-1;i++)
     cout<<v[i]<<", ";
 return 0;}
```

Rulare
Dacă iniţial datele citite au fost: n=1,
v[4] este 9 -2 4 1
paşii rulaţi sunt
i=0
min=v[i]=v[0]=9, poz=i=0
j=1 v[1]<9, adevărat, min=-2, poz=1
j=2 v[2]<-2, fals
j=3 v[3]<-2, fals
v[0]<->v[1] vectorul va fi -2 9 4 1
i=1
min=v[i]=v[1]=9, poz=i=1
j=2 v[2]<9, adevărat, min=4, poz=2
j=3 v[3]<4, adevărat, min=1, poz=3
v[1]<->v[3] vectorul va fi -2 1 4 9
i=2
min=v[i]=v[2]=4, poz=i=2
j=3 v[3]<4, fals
v[2]<->v[2]
Se termină bucla repetitivă şi se afişează conţinutul vectorului ordonat -2 1 4 9

Probleme propuse

1. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector şi un număr nr. Să se afişeze poziţiile pe care se află în vector sau un mesaj în cazul în care nu se găseşte.

2. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector şi se afişează elementul maxim dintre ele.

3. Scrieţi un program în C++ în care se citesc elementele a doi vectori A[ ] şi B[ ] cu n respectiv m elemente întregi. Să se completeze vectorul C[ ] cu elementele vectorului A[ ], urmate de elementele vectorului B[ ] după care să se afişeze conţinutul vectorului C[ ].

4. Scrieţi un program în C++ în care se citesc elementele unui vector cu 0<n<10 şi 0<V[i]<10. Să se formeze cu cifrele vectorului un număr şi să se afişeze.

5. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector. Să se afişeze dacă sunt distribuite simetric sau nu faţă de mijlocul vectorului. Dacă vectorul conţine 4, 1, 4 sau -5, 7, 7, -5 este simetric.

6. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector. Să se afişeze elementele prime de pe poziţii pare din vector.

7. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector, un număr întreg nr şi un număr pozitiv k<n. Să se insereze numărul nr pe poziţia k şi să se afişeze vectorul.

8. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector şi un număr pozitiv k<n. Să se şteargă numărul de pe poziţia k şi să se afişeze vectorul. Se va rezolva prin deplasarea elementelor vectorului.

9. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector. Să se găsească elementul maxim din vector, să se înlocuiască cu pătratul lui şi să se afişeze vectorul. Dacă maximul se găseşte pe mai multe poziţii în vector, să se modifice peste tot.

10. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector şi se afişează primele x elemente în ordine crescătoare şi următoarele în ordine descrescătoare.

11. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector, se ordonează descrescător şi se afişează vectorul ordonat.

12. Scrieţi un program în C++ în care se citesc n elemente întregi ale unui vector şi se afişează pe un rând elementele pare şi pe alt rând elementele impare din vector.

13. Scrieţi un program în C++ în care se citeşte un vector cu n elemente reale. Să se afişeze dacă elementele vectorului sunt distincte sau nu.

14. Scrieţi un program în C++ în care se citeşte un număr întreg pozitiv x şi se afişează numărul în baza 2. Dacă se citeşte 6 se afişează 110, pentru 13 se afişează 1101.

15. * Scrieţi un program în C++ în care se citeşte un număr real x şi un vector ce conţine coeficienţii unei ecuaţii de gradul n, cu 0<n<11. Să se afişeze valoare polinomului în punctul x.

16. Scrieţi un program în C++ în care se citesc elementele a doi vectori A şi B. Să se afişeze:
a) $A \cup B$. Ştim că reuniunea a două mulţimi înseamnă toate elementele lui A şi lui B sunt luate o singură dată
b) $A \cap B$. Intersecţia lui A cu B înseamnă că rezultatul conţine numai elementele comune lui A şi lui B.
c) A-B. Diferenţa A-B conţine elementele care aparţin lui A şi nu aparţin lui B.

17. Se citeşte un vector cu n elemente întregi. Să se afişeze vectorul permutat circular spre dreapta.

18. ** Definim noţiunea de pereche ordonată, perechea de numere naturale (x,y) cu $x \leq y$. Definim cel mai mic multiplu comun al unei perechi ordonate ca fiind cel mai mic multiplu comun al numerelor care formează perechea.
Se dau k numere naturale n1, n2, … , nk.

Cerinţă:
Să se determine pentru fiecare dintre numerele ni (i=1,2,…,k):
a) câte perechi ordonate au cel mai mic multiplu comun egal cu ni.
b) dintre acestea, perechea ordonată care are suma minimă.
Date de intrare:
Prima linie a fişierului cmmmc.in conţine un număr natural k. Următoarele k linii din acest fişier vor conţine câte un număr natural; linia i+1 va conţine numărul ni (i=1,2,…,k) .
Date de ieşire:

Fişierul cmmmc.out va conţine k linii. Pe fiecare dintre acestea se vor afla trei numere. Cele trei numere de pe linia i vor reprezenta:
- primul, numărul de perechi ordonate care au cel mai mic multiplu comun egal cu ni;
- următoarele două, numerele care alcătuiesc perechea ordonată care are cel mai mic multiplu comun egal cu ni şi a căror sumă este minimă, afişate în ordine crescătoare.

Restricţii:

- $1 \leq k \leq 100$
- $1 \leq ni \leq 2\ 000\ 000\ 000$
- Pentru 20% dintre teste, $k \leq 100$ şi $ni \leq 1000$

Exemple:

cmmmc.in	cmmmc.out	Explicaţii
2 10 11	5 2 5 2 1 11	Există cinci perechi distincte care au cel mai mic multiplu comun egal cu 10: (1,10), (2,10), (5,10), (2,5) (10,10). Dintre acestea perechea cu cea mai mică sumă este (2,5). Pentru n=11 există două perechi ordonate care au cel mai mic multiplu comun 11: (1,11), (11,11). Dintre acestea perechea cu cea mai mică sumă este (1,11).

(ONI 2010)

19. *** Gina şi Mihai joacă împreună jocul Cuart. Ei au la dispoziție un șir de 2*N cartonașe ce conțin numere naturale. Primele N cartonașe, de la stânga la dreapta, sunt ale Ginei, iar următoarele N ale lui Mihai. Gina traversează șirul, de la stânga la dreapta și scrie pe o foaie de hârtie, pe primul rând, un șir de numere obținut din numerele de pe cartonașele sale, din care a șters toate cifrele pare. La fel procedează Mihai care scrie pe foaia sa de hârtie, pe primul rând, șirul de numere obținut din numerele de pe cartonașele sale, din care a șters toate cifrele impare. Dacă dintr-un număr s-au șters toate cifrele, sau au rămas doar cifre egale cu 0, atunci numărul este ignorat, deci pe hârtie nu se scrie nimic.
Fiecare copil, notează pe hârtia sa, pe al doilea rând, un alt șir de numere obținut astfel: pentru fiecare număr X scris pe primul rând, copilul va scrie cel mai mare număr natural K cu proprietatea că $1+5+9+13+...+ K \leq X$. În jocul copiilor, numărul X se numește cuarț dacă $1+5+9+13+...+K = X$.

Cartonașele Ginei				Cartonașele lui Mihai			
1234	48	284260	75	756	1232515	153	98
Numerele scrise de Gina				Numerele scrise de Mihai			
13	75			6	22	8	
5	21			5	9	5	

În exemplul de mai sus, Gina nu a scris niciun număr cuarț pe primul rând, iar Mihai a scris unul singur (6=1+5).
Regulile de câștig ale jocului sunt următoarele:

- Câștigă acel copil care are scrise pe primul rând cele mai multe numere cuarț. În acest caz, valoarea de câștig a jocului este egală cu numărul de numere cuarț scrise de copilul câștigător.
- Dacă cei doi copii au scris același număr de numere cuarț, atunci va câștiga cel care are primul număr scris pe primul rând, mai mare decât al celuilalt. Acest prim număr scris de câștigător va reprezenta valoarea de câștig.
- Dacă nici Gina și nici Mihai nu au scris niciun număr pe hârtie, se consideră egalitate și nu câștigă niciunul.

Cerinţe:
Scrieţi un program care să citească numărul N reprezentând numărul de cartonașe ale unui copil şi cele 2*N numere de pe cartonașe, în ordine de la stânga la dreapta și care să determine:
1) Cel mai mare număr de pe cele 2*N catonașe, pentru care nu s-a scris niciun număr pe primul rând (a fost omis), nici pe hârtia Ginei, nici pe hârtia lui Mihai; dacă nu a fost omis niciun număr, se va scrie 0;
2) Câștigătorul jocului și afișează numărul 1 dacă a câștigat Gina, 2 pentru Mihai sau 0 în caz de egalitate.
3) Valoarea de câștig a jocului, sau 0, în caz de egalitate.

Date de intrare:
Fişierul de intrare cuart.in conţine pe prima linie un număr natural P. Pentru toate testele de intrare, numărul P poate avea doar valoarea 1, valoarea 2 sau valoarea 3. Pe a doua linie a fișierului de intrare cuart.in se găsește numărul natural N reprezentând numărul de cartonașe ale fiecărui copil și pe a treia linie, în ordine de la stânga la dreapta, numerele de pe cele 2*N cartonașe, separate prin câte un spațiu.
Date de ieşire:
Dacă valoarea lui P este 1, se va rezolva numai punctul 1) din cerințe. În acest caz, fișierul de ieșire cuart.out va conține pe prima linie un număr natural reprezentând răspunsul la cerinţa 1).
Dacă valoarea lui P este 2, se va rezolva numai punctul 2) din cerințe. În acest caz, fișierul de ieșire cuart.out va conține pe prima linie un număr natural reprezentând răspunsul la cerinţa 2).
Dacă valoarea lui P este 3, se va rezolva numai punctul 3) din cerințe. În acest caz, fișierul de ieșire cuart.out va conține pe prima linie un număr natural reprezentând răspunsul la cerinţa 3).

Restricţii:

- $1 \leq N \leq 1000$

- 1 ≤ numerele de pe cartonașe < 100000000
- Pentru rezolvarea corectă a primei cerinţe se acordă 20 de puncte, pentru rezolvarea corectă a celei de a doua cerințe se acordă 30 de puncte, pentru rezolvarea corectă a celei de a treia cerințe se acordă 50 de puncte.

Exemple:

cuart.in	**cuart.out**	**Explicaţii**
1 4 1234 48 284260 75 756 1232515 153 98	284260	P = 1, pentru acest test, se rezolvă cerința 1). Gina a scris pe hârtia sa, pe două rânduri numerele: 13 75 5 21 Mihai a scris pe hârtie numerele: 6 22 8 5 9 5 Cel mai mare număr omis este 284260

cuart.in	**cuart.out**	**Explicaţii**
2 4 1234 48 284260 75 756 1232515 153 98	2	P = 2, pentru acest test, se rezolvă cerința 2). A câștigat Mihai deoarece are un număr cuarț, iar Gina niciunul

cuart.in	**cuart.out**	**Explicaţii**
3 1 154 2181	28	P = 3, pentru acest test, se rezolvă cerința 3). Gina a scris pe hârtia sa, pe două rânduri numerele: 15 9 Mihai a scris pe hârtie numerele: 28 13 Ambii copii au scris câte un număr cuarț, însă a câștigat Mihai care are primul număr scris pe primul rând mai mare decât al Ginei. Valoarea de câștig a jocului este 28.

(OJI 2015, clasa a V-a)

Capitolul 2.

Tablouri bidimensionale (matrice)

Noțiuni teoretice

Un tablou bidimensional (matrice) este o structură de elemente de acelaşi tip plasate în locaţii de memorie alăturate şi care pot fi accesate cu ajutorul unor indici cu două dimensiuni.

Am putea spune că o matrice este un „vector de vectori”.

Dacă pentru un vector aveam nevoie de n pentru numărul de elemente, la o matrice avem nevoie de n pentru numărul de linii şi de m pentru numărul de coloane.

De exemplu, mai jos avem o matrice, pe care o notăm cu mat, care are 3 linii şi 4 coloane:

5	-2	8	27
1	0	5	7
-6	15	19	8

mat

În exemplul dat avem:
mat[0][0]=5, mat[0][1]= -2, mat[0][2]=8, mat[0][3]=27,
mat[1][0]=1, mat[1][1]=0, mat[1][2]=5, mat[1][3]=7
mat[2][0]= -6, mat[2][1]=15, mat[2][2]=19, mat[2][3]=8

O matrice se defineşte prin numărul de linii n, numărul de coloane m şi elementele mat[i][j] unde $i \in \{1, 2, ..n\}$ şi $j \in \{1, 2, ..m\}$

Probleme rezolvate

Problema 1. Scrieţi un program în C++ în care se citesc numărul de linii, numărul de coloane şi elementele întregi ale unei matrice. Să se afişeze conţinutul matricei.

#include <iostream>
using namespace std;
int main()
{int mat[10][10], n, m, i, j;
cout<<"n=";cin>>n;
cout<<"m=";cin>>m;
for(i=0;i<=n-1;i++)
for(j=0;j<=m-1;j++)
{cout<<"mat["<<i<<"]["<<j<<"]=";
cin>>mat[i][j]; }
for(i=0;i<=n-1;i++)
{for(j=0;j<=m-1;j++)
cout<<mat[i][j]<<" ";
cout<<'\n';}
return 0;}
	se citeşte n, numărul de linii ale matricii
	se citeşte m, numărul de coloane ale matricii
	
	se citesc elementele matricii
	
	se afişează elementele matricii, cu spaţiu între ele, pe linii diferite

Problema 2. Scrieţi un program în C++ în care afişează elementele de pe diagonala principală şi de pe diagonala secundară a unei matrici, pe linii diferite.

#include <iostream>
using namespace std;
int main()
{int mat[10][10], n, i, j;
cout<<"n=";cin>>n;
for(i=0;i<=n-1;i++)
for(j=0;j<=n-1;j++)
cin>>mat[i][j];
for(i=0;i<=n-1;i++)
cout<<mat[i][i]<<" ";
cout<<'\n';
for(i=n-1;i>=0;i--)
cout<<mat[i][2-i]<<" ";
return 0;}
	Presupunem că matricea citită este
	1 2 3
	4 5 6
	7 8 9
	Trebuie să se afişeze:
	1 5 9
	7 5 3
	
	Observaţie:
	elementele de pe diagonala principală au proprietatea că i=j
	Observaţie:
	Diagonalele principală şi secundară se calculează pentru o matrice pătratică

Problema 3. Scrieţi un program în C++ în care se interschimbă conţinutul a două linii x şi y unde 0<x<n şi 0<y<n.

#include <iostream> using namespace std; int main() {int mat[10][10], n, m, i, j, x, y, aux; cin>>n>>m>>x>>y;	

for(i=0;i<=n-1;i++)
  for(j=0;j<=m-1;j++)
  cin>>mat[i][j];
for(j=0;j<=n-1;j++)
  {aux=mat[x][j];
  mat[x][j]=mat[y][j];
  mat[y][j]=aux;}
for(i=0;i<=n-1;i++)
  {for(j=0;j<=m-1;j++)
  cout<<mat[i][j]<<" ";
  cout<<'\n';}
return 0;}
	se interschimbă conţinutul liniilor x şi y
	
	se afişează elementele matricii cu spaţii între ele, pe linii diferite

Problema 4. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează transpusa ei.

#include<iostream>
using namespace std;
int main()
{int mat[10][10], n, m, i, j;
  cout<<"n=";cin>>n;
  cout<<"m=";cin>>m;
  for(i=0;i<=n-1;i++)
  for(j=0;j<=m-1;j++)
  {cout<<"mat["<<i<<"]["<<j<<"]=";cin>>mat[i][j];}
  for(i=0;i<=m-1;i++)
  { for(j=0;j<=n-1;j++)
  cout<<mat[j][i]<<" ";
  cout<<'\n'; }
return 0;}
	matricea iniţială
	1 2 3
	4 5 6
	7 8 9
	5 7 2
	
	transpusa matricii
	1 4 7 5
	2 5 8 7
	3 6 9 2

Problema 5. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează suma elementelor de pe marginea ei.

```
#include<iostream>
using namespace std;
int main()
{int mat[10][10], n, m, i, j, s=0;
  cout<<"n=";cin>>n;
  cout<<"m=";cin>>m;
```

```
        for(i=0;i<=n-1;i++)
          for(j=0;j<=m-1;j++)
            {cout<<"mat["<<i<<"]["<<j<<"]=";cin>>mat[i][j];}
        for(i=0;i<=n-1;i++)
         s=s+mat[i][0]+mat[i][m-1];
         for(j=1;j<=m-2;j++)
            s=s+mat[0][j]+mat[n-1][j];
         cout<<s;
      return 0;}
```

Problema 6. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează matricea permutată circular în sus.

```
#include<iostream>
using namespace std;
int main()
{ int n,m,i,j,t,mat[10][10];
   cout<<"n=";cin>>n;
   cout<<"m=";cin>>m;
   for(i=0;i<=n-1;i++)
    for(j=0;j<=m-1;j++)
     {cout<<"mat["<<i<<"]["<<j<<"]=";
      cin>>mat[i][j];  }
   for(t=0;t<=n-1;t++)
   {
   for(j=0;j<=m-1;j++)
     mat[n][j]=mat[0][j];
   for(i=0;i<=n-1;i++)
    for(j=0;j<=n-1;j++)
     mat[i][j]=mat[i+1][j];
   for(i=0;i<=n-1;i++)
    { for(j=0;j<=m-1;j++)
       cout<<mat[i][j]<<" ";
     cout<<'\n';
    }
    cout<<'\n';
   }
   return 0;}
```

```
matricea iniţială pentru n=2 şi m=2
1 2
3 4
5 6


afişare
3 4
5 6
1 2

5 6
1 2
3 4

1 2
3 4
5 6
```

Problema 7. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane, $0 \le n \le 9$, $0 \le m \le 9$ şi se afişează mulţimea formată din componentele matricei, aflate pe linii şi coloane pare.

```
#include<iostream>
using namespace std;
int main()
{ int n,m,i,j,t,mat[10][10], v[100], k, ok;
  cout<<"n=";cin>>n;
```

```
matricea iniţială pentru n=4 şi m=5
5 0 1 4 5
6 1 7 6 8
3 2 1 4 2
4 1 2 6 5
```

```
  cout<<"m=";cin>>m;
  for(i=0;i<=n-1;i++)
   for(j=0;j<=m-1;j++)
   {cout<<"mat["<<i<<"]["<<j<<"]=";
    cin>>mat[i][j];}
    k=0;
   for(i=0;i<=n-1;i=i+2)
    for(j=0;j<=m-1;j=j+2)
      {ok=1;
       for(t=0;t<=k;t++)
         if(mat[i][j]==v[t])
             ok=0;
        if(ok==1)
        {v[k]=mat[i][j];
         k++;}
      }
   for(i=0;i<=k-1;i++)
      cout<<v[i]<<" ";
return 0;}
```

se afişează 5 3 1 2

Problema 8. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane, $0 \le n \le 9$, $0 \le m \le 9$, se şterge linia care începe cu elementul maxim de pe linia respectivă şi se afişează matricea rezultată.

```
#include<iostream>
using namespace std;
int main()
{ int n,m,i,j,mat[10][10], max, ok, r, c;
  cout<<"n=";cin>>n;
  cout<<"m=";cin>>m;
  for(i=0;i<=n-1;i++)
   for(j=0;j<=m-1;j++)
   {cout<<"mat["<<i<<"]["<<j<<"]=";
    cin>>mat[i][j];}
     i=0;
     while(i<=n-1)
      {max=mat[i][0];ok=1;
         for(j=1;j<=m-1;j++)
          if(mat[i][j]>max)
            {max=mat[i][j];
             ok=0;}
       if(ok==1)
        {
         for(r=i; r<=n-2;r++)
          for(c=0;c<=m-1;c++)
          mat[r][c]=mat[r+1][c];
          n=n-1;
        }
```

```
matricea iniţială pentru n=4 şi m=5
2 3 9 4 1
7 5 4 1 2
8 4 1 2 3
5 6 7 8 4

afişare
2 3 9 4 1
5 6 7 8 4
```

```
        else
          i=i+1;
      }
 for(i=0;i<=n-1;i++)
   {for(j=0;j<=m-1;j++)
      cout<<mat[i][j]<<" ";
      cout<<'\n';}
return 0;}
```

Probleme propuse

1. Scrieţi un program în C++ în care se citesc elementele întregi ale unei matrice cu n linii şi m coloane şi se afişează media aritmetică a elementelor matricei.

2. Scrieţi un program în C++ în care se citesc elementele întregi ale unei matrice cu n linii şi m coloane şi se afişează minimul de pe fiecare linie a matricei.

3. Scrieţi un program în C++ în care se citesc elementele întregi ale unei matrice cu n linii şi m coloane şi se afişează suma elementelor de pe fiecare linie de ordin par a matricei.

4. Scrieţi un program în C++ în care se citesc elementele întregi ale unei matrice cu n linii şi m coloane şi se afişează suma elementelor de pe diagonala principală şi produsul elementelor diferite de zero de pe diagonala secundară.

5. Scrieţi un program în C++ în care se citesc elementele întregi ale unei matrice cu n linii şi m coloane şi se afişează elementele palindrom de sub diagonala principală şi un mesaj dacă nu sunt elemente palindrom.

6. Scrieţi un program în C++ în care se interschimbă conţinutul a două coloane x şi y ale unei matrice, unde 0<x<m şi 0<y<m.

7. Construiţi şi afişaţi matricele după modelele de mai jos, pentru un număr întreg n citit de la tastatură:

a)	b)	c)	d)
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4	* / / / / * / / / / * / / / / *	0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0	1 0 0 0 1 3 1 0 1 4 3 3 1 4 4 3 1 5 1 4 1 5 5 5 1

8. Scrieţi un program în C++ în care se citesc două matrice A şi B cu n linii şi m coloane şi se afişează transpusa diferenţei A-B.

9. Scrieţi un program în C++ în care se citeşte o matrice pătratică, se ordonează elementele de pe diagonala secundară şi se afişează matricea rezultată.

10. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează matricea cu elementele ordonate crescător pe coloane.

11. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane, se construieşte şi se afişează un vector cu elementele matricei.

12. * Construiţi o matrice pătratică cu primii x^2 termeni ai şirului lui Fibonacci.

13. * Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează elementele matricei în spirală începând cu mat[0][0].

14. Scrieţi un program în C++ în care se citeşte o matrice pătratică. Să se afişeze dacă este simetrică faţă de:
a) diagonala principală; b) diagonala secundară;
c) linia mediană; d) coloana mediană;

15. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane şi se afişează matricea permutată circular de la stânga la dreapta.

16. Scrieţi un program în C++ în care se citeşte o matrice cu n linii şi m coloane, $0 \le n \le 9$, $0 \le m \le 9$ şi se afişează mulţimea formată din componentele matricei, aflate pe cele două linii paralele cu diagonala secundară.

17. ** Problema lasere

Se consideră un teren reprezentat printr-o matrice cu n linii şi n coloane având elemente numere naturale. În fiecare element al matricei este memorată înălţimea zonei de teren corespunzătoare ca poziţie elementului respectiv. Pe acest teren sunt amplasate m lasere, în poziţii cunoscute. Un laser este îndreptat spre unul dintre cele 4 puncte cardinale, codificate prin numere astfel: Nord prin valoarea 1, Est prin valoarea 2, Sud prin valoarea 3 şi respectiv Vest prin valoarea 4. Fiecare laser va executa o singură tragere şi ca urmare va scădea cu 1 valorile tuturor elementelor din matrice din direcţia sa de tragere, exceptând poziţia laserului respectiv.

După efectuarea tuturor tragerilor, se caută poziţiile tuturor gropilor şi ale tranşeelor.

Numim *groapă* un element din matrice pentru care toate cele 8 elemente învecinate pe linie, coloană sau diagonale au valori mai mari sau egale decât el.

Numim *tranşee* o secvenţă maximală formată din două sau mai multe gropi situate pe aceeaşi linie, pe coloane consecutive. Secvenţa se numeşte maximală dacă nu mai poate fi prelungită la niciunul dintre capete.

Cerinţe:
Cunoscând configuraţia terenului şi amplasarea laserelor, să se rezolve una dintre următoarele două cerinţe:

1. să se determine numărul de gropi din teren, după executarea tragerilor;
2. să se determine numărul de tranşee existente, după executarea tragerilor.

Date de intrare:
Fişierul de intrare lasere.in conţine pe prima linie un număr natural c care reprezintă cerinţa ce urmează să fie rezolvată (1 sau 2). Pe a doua linie se află două numere naturale n şi m, reprezentând numărul de linii şi de coloane ale matricei, respectiv numărul de lasere. Pe următoarele n linii se află câte n numere naturale, reprezentând elementele matricei. Pe următoarele m linii sunt descrise cele m lasere, câte un laser pe o linie. Pe o linie care descrie un laser se află 3 numere naturale i j d, cu semnificaţia că se află un laser pe linia i şi coloana j ($1 \le i,j \le n$), care trage în direcţia d ($1 \le d \le 4$). Valorile situate pe aceeaşi linie sunt separate prin spaţiu.

Date de ieşire:
Fişierul de ieşire lasere.out va conţine pe prima linie un singur număr natural. Acest număr reprezintă numărul de gropi (dacă c=1) sau numărul de tranşee (dacă c=2).

Restricţii şi precizări:

- $4 \le n \le 200$
- $1 \le m \le 200$
- Numerotarea liniilor şi a coloanelor este de la 1 la n.
- Elementele matricei din fişierul de intrare sunt numere naturale de maxim 4 cifre.
- Poziţiile laserelor sunt distincte.
- Pentru teste valorând 30% din punctaj cerinţa este 1.

Exemple:

lasere.in	**lasere.out**	**lasere.in**	**lasere.out**	**Explicaţii**
1 5 3 1 1 3 **4** 5 8 7 **6** 5 4 9 3 5 6 7 1 1 1 **9** 8	6	2 5 3 1 1 3 **4** 5 8 7 **6** 5 4 9 3 5 6 7 1 1 1 **9** 8	1	După ce acţionează laserele terenul arată astfel: 1 1 3 4 4 8 7 6 5 4 9 3 4 6 7 0 0 -1 9 8

1 1 1 5 6 2 3 3 4 4 4 1 4 2		1 1 1 5 6 2 3 3 4 4 4 1 4 2		1 1 0 5 6 Există 6 gropi şi o tranşee. Se număra gropile chiar dacă fac parte dintr-o tranşee.

(OJI 2015, clasa a VII-a)

Capitolul 3.

Șiruri de caractere

Noțiuni teoretice

Numim șir de caractere o succesiune de caractere așezate unul după altul.

Exemple de șiruri de caractere: „informatica”; „programarea calculatoarelor”; „Ce filme ai vizionat în 2015?”

În memoria internă a calculatorului, șirul de caractere se reprezintă ca o succesiune de coduri ASCII, corespunzătoare caracterelor componente.

ASCII este acronimul de la American Standard Code for Information Interchange. Tabela ASCII normală conține codurile pentru 128 caractere.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Tabela ASCII extinsă conține codurile pentru încă 128 caractere.

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	·
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

În Tabela ASCII extinsă am dat codurile caracterelor în baza 10 și sunt aranjate pe coloane (cod caracter).

De obicei, codurile caracterelor din tabela ASCII se dau în patru baze de numerație: baza 16 (hexazecimal), baza 10 (decimal), baza 8 (octal), baza 2 (binar).

Observații:

- cifrele încep cu codul ASCII 48(decimal) și sunt așezate una după alta;
- literele mari încep cu codul ASCII 65(decimal) și sunt așezate una după alta;
- literele mici încep cu codul ASCII 97(decimal) și sunt așezate una după alta.

Vedeți Problema 1 de la Probleme rezolvate.

În limbajul C++ șirul de caractere începe cu poziția de index 0 (zero).
Orice șir de caractere se termină cu caracterul NULL(cod ASCII 0).

Exemple:

1) Șirul „Internet” are forma următoare:

0	1	2	3	4	5	6	7	8
I	n	t	e	r	n	e	t	NULL

și în memoria internă are următoarele coduri ASCII (baza 10):

73	110	116	101	114	110	101	116	0

2) Șirul „Drum bun !” are forma următoare:

0	1	2	3	4	5	6	7	8	9	10

D	r	u	m		b	u	n		!	NULL

și în memoria internă are următoarele coduri ASCII(baza 10):

68	114	117	109	32	98	117	110	32	33	0

Un șir de caractere poate fi tratat (citit, prelucrat, scris) în două moduri:

- ca vector de caractere, deci caracter cu caracter;
- ca șir propriu-zis de caractere, folosind funcții specifice.

Vedeți Problemele 2 și 3 de la Probleme rezolvate.

Inițializare șir de caractere

Limbajul C/C++ permite inițializarea unui tablou de caractere cu câte un caracter pe fiecare poziție. De asemenea, un șir poate fi inițializat printr-o constantă șir, care include automat caracterul NULL pe ultima poziție.

Exemple:

```
char tablou[5] = {'a', 'v', 'i', 'o', 'n'};  // tablou de caractere; nu are NULL
char sir[100]="avion";   //șir de caractere; are NULL
```

Citirea și afișarea șirurilor de caractere:

1) Folosind funcția cin>> din header-ul < iostream>

Caracterul NULL este adăugat automat. Dezavantaj: nu se pot citi șiruri care conțin mai multe cuvinte separate prin spații (blank, space) sau tab(salt peste coloane). Citirea șirului se sfârșește la întâlnirea primului caracter spațiu sau tab.

Exemplu:

```
char s[30];
cin>>s; //dacă se citește "ora de informatica", variabila s va reține numai "ora"
cout<<s; //afișează șirul "ora"
```

2) Folosind funcția cin.get() din header-ul <string.h>

Exemplu:

```
char s[30], x;  int n;
cin.get(s,n,x);
```

Funcția cin.get citește un șir de caractere s în una din situațiile:

- până când au fost citite n-1 caractere,
- până s-a întâlnit caracterul x,
- până s-a întâlnit caracterul '\n' (new line).

Sunt citite și caracterele albe, caracterul NULL este inserat automat la sfârșitul șirului, iar caracterul x transmis ca ultim parametru nu este inserat în șir.

Un caracter alb (whitespace) este un caracter ASCII care poate fi vizualizat pe ecran sub forma unui spațiu liber. Exemple: spaţiu (space, blank, cod ASCII

32), sfârşit de linie (’\n’ new line, cod ASCII 10), tab (cod ASCII 9), pagină nouă(new page, cod ASCII 12).

Exemple:

```
char a[30];
cin.get(a,5,’s’);       //dacă se citește șirul “gimnaziu”, variabila a va reține “gimn”
cin.get(a,15,’n’);      // dacă se citește șirul “gimnaziu”, variabila a va reține “gim”
cin.get(a,10);  // dacă se citește șirul “gimnaziu”, variabila a va reține “gimnaziu”
```

Funcția cin.get( ) fără parametri are rolul de a citi un caracter (alb sau nu).

Funcția cin.get(char c) are rolul de a citi un caracter (alb sau nu) pe care îl încarcă în variabila c.

Observație: Dacă utilizăm repetat funcția cin.get(a,nr,x), după fiecare folosire trebuie citit caracterul de la sfârșitul fiecărui șir , adica ’\n’. În caz contrar, acest caracter va fi încărcat la începutul următorului șir, iar al doilea șir va fi vid. Citirea caracterului ’\n’ se realizează folosind cin.get() fără parametri.

Exemplu:

```
char a[30],b[30];
cin.get(a,15); cout<<a;
cin.get(b,10);cout<<b;
```

Dacă se încearcă citirea sirurilor „programare” și „calculatoare”, se observă ca a=”programare”, b=”” (nici nu apucăm să citim șirul b).

Varianta corectă este:

```
cin.get(a,15); cout<<a;
cin.get();
cin.get(b,10); cout<<b;
```

Vedeți Problema 4 de la Probleme rezolvate.

Funcții uzuale de lucru cu șiruri de caractere din header-ul <string.h>

- Funcția **strlen**

int **strlen**(nume_sir); – returnează lungimea efectivă a unui șir.

Exemplu:

```
char a[50]= "hai la mare"; - strlen(a) = 11
```

- Funcția **strcpy**

strcpy(sir_destinatie,sir_sursa); **–** copiază șirul sir_ sursa în sir_destinatie (se simulează atribuirea a=b).

Nu este permisă, în limbajul C++, atribuirea între două șiruri de caractere folosind operatorul =.

Exemplu:

```
char a[50]= "salut azi",b[40]= "buna ziua";
```

a=b; //eroare
strcpy(a,b); - a = "buna ziua"; b="buna ziua";

- Funcția **strcat**

strcat(dest,sursa); – adaugă șirului dest șirul sursa.
Șirul sursa rămâne nemodificat. Operația se numește *concatenare*.

Exemplu:

char a[5]= "hai ",b[26]= "la mare departare! ";
strcat(a,b); -a = "hai la mare departare! ";

- Funcția **strchr**

d=strchr(sir,c); – caută caracterul c în șirul sir. Căutarea se face de la stânga la dreapta; returnează în d adresa subșirului care începe cu prima apariție a caracterului c. Dacă nu este găsit caracterul, funcția returnează 0. Diferența dintre adresa subșirului returnat d și cea a șirului inițial a reprezintă chiar poziția caracterului căutat în șirul dat.

Exemplu:

char a[40]= "sa invatam programare",b='t',c='x',*d;
cout<<strchr(a,b); - se tipărește "tam programare";
cout<<strchr(a,c); - nu se tipărește nimic
d= strchr(a,b);cout<<"Caracterul apare prima data la pozitia "<<d-a;
Vedeți Problema 5 de la Probleme rezolvate.

- Funcția **strcmp**

int strcmp(sir1,sir2); – are rolul de a compara două șiruri de caractere.
Valoarea returnată este <0 (dacă sir1<sir2), =0 (dacă sir1=sir2) și >0 (dacă sir1>sir2). Funcția strcmp face deosebire între literele mari și cele mici ale alfabetului.

Exemplu:

char sir1[20]= "La plimbare", sir2[15]= "La preumblare";
int dif=strcmp(sir1,sir2); - variabila dif<0 deorece caracterul 'l' din sir1 este mai mic decât caracterul 'r' din sir2 (vezi în tabela ASCII).

- Funcția **strstr**

strstr(sir1,sir2); – are rolul de a identifica dacă șirul sir2 este subșir al șirului sir1. Dacă este, funcția returnează adresa de început a primei apariții a subșirului sir2 în șirul sir1, altfel returnează adresa 0. Căutarea se face de la stânga la dreapta. Diferența dintre adresa găsită a subșirului sir2 și cea a șirului sir1 reprezintă chiar poziția apariției lui sir2 în sir1.

Exemplul 1:

char a[30]= "citestc carti multe", b[20]= "carti", *d;

d=strstr(a,b);
cout<<"Sirul b apare in sirul a in pozitia "<<d-a; // afișează poziția 7

Exemplul 2:

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
	char a[30]= "citestc carti multe", b[20]= "carti", *d;
	d=strstr(a,b);
	if (d!=0) cout<<"Sirul b apare in sirul a in pozitia"<<d-a;
			else cout<<"Sirul b nu se gaseste in sirul a";

	 return 0;
}
```

- Funcția **strlwr**

strlwr(sir); – are rolul de a converti toate literele mari din sir în litere mici. Restul caracterelor din sir rămân neschimbate.

Exemplul 1:

char sir[30]= "Azi esTe o VreMe buna!! ";
strlwr(sir); //se obține șirul "azi este o vreme buna!! "

Exemplul 2:

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
   char sir[30];
   cout << "Tastati sirul dorit: "; cin.get(sir,30);
   strlwr(sir);
   cout<<"Sirul cu litere mici este: "<<sir;
   return 0;
}
```

- Funcția **strupr**

strupr(sir); – are rolul de a converti toate literele mici din sir în litere mari. Restul caracterelor din sir rămân neschimbate.
Exemplu: vedeți Problema 6 de la Probleme rezolvate.

Funcții uzuale de lucru cu șiruri de caractere din header-ul <stdlib.h>

- Funcția **atof**

double atof(sir); – convertește un șir către tipul double. Dacă eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0. În șir se acceptă caracterul separator . (punct) între partea întreagă și partea zecimală.

Exemplu:

```
#include <iostream>
#include <stdlib.h>
// header-ul stdlib.h contine functiile de conversie intre siruri si numere
using namespace std;
int main()
{
   char sir[20]; double numar;
   cout << "Tastati sirul de convertit: "; cin.get(sir,20);
   numar=atof(sir);
   if (numar!=0) cout<<"Numarul obtinut este: "<<numar;
        else cout<<"Eroare! Caracter nenumeric in sir.";
   return 0;
}
```

- Funcția **atoi**

int atoi(sir); – convertește un șir către tipul int. Dacă eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.

- Funcția **atol**

long atol(sir); – convertește un șir către tipul long. Dacă eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.

- Funcția **itoa**

itoa(int valoare,sir,int baza); – convertește o valoare de tip int în șir, care este memorat în variabila sir. Parametrul baza indică baza de numerație către care se face conversia.

Exemplul 1:

```
int nr=213, baza=8; char sir[17];
itoa(nr,sir,baza); //în sir se obține “325”, care este nr în baza 8
```

Exemplul 2:

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{
   int nr, baza; char sir[17];
   cout << "Tastati numarul de tip intreg: ";cin>>nr;
   cout<< "Care este baza catre care se face conversia?: "; cin>>baza;
   itoa(nr,sir,baza);
```

```
    cout<<"S-a obtinut sirul: "<<sir<<" iar baza este "<<baza<<endl;
    return 0;
  }
```

- Funcția **ltoa**

ltoa(long valoare,sir,int baza); – convertește o valoare de tip long int în șir, care este memorat în variabila sir.

Nu am tratat toate funcțiile pentru șiruri de carctere, restul rămânând spre studiul celor curioși! Succes!

Probleme rezolvate

Problema 1.

Scrieți în limbajul C++ un program care realizează următoarele:

- afișează toate codurile tipăribile din tabela ASCII normală, pe 9 coloane ale monitorului;
- afișează toate codurile tipăribile din tabela ASCII extinsă, pe 7 coloane ale monitorului.

Pentru fiecare caracter se vor tipări: codul ASCII(în baza 10) urmat de caracterul asociat.

```
#include <iostream>
#include<iomanip>  // header-ul ce contine functia setw()
using namespace std;
int main()
{
   cout << "Caracterele tiparibile din tabela ASCII normala sunt:" << endl;
   for(int i=32;i<=126;i++) //codurile ASCII pt. caracterele tiparibile
         {cout<< setw(6)<<i<<" "<<char(i);
         //afisam exact pe 6 pozitii: codul ASCII, un spatiu si caracterul asociat
          if ((i-31)%9==0) cout<<endl;
          //afisam pe 9 coloane;
          //dupa fiecare a 9-a coloana trecem la linia urmatoare pe ecran
         }
   cout<<endl<<endl;
   cout << "Caracterele tiparibile din tabela ASCII extinsa sunt:" << endl;
   for(int i=128;i<=255;i++)
      {cout<< setw(6)<<i<<" "<<char(i);
       if ((i-127)%7==0) cout<<endl;
      }
   return 0;
}
```

Problema 2. Citim de la tastatură un șir de caractere format dintr-un singur cuvânt (nu conține spații). Să se afișeze lungimea șirului și literele mici care intră în componența lui.

Exemplu: Dacă vom tasta cuvântul "LibErtaTe!!", atunci se va afișa lungimea 11 și literele mici "ibrtae".

Rezolvare cu vector:

```
#include <iostream>
using namespace std;
int main()
{   char cuv[20];
   int i,lung;
   cout << "Tastati cuvantul: "; cin>>cuv;
   lung=0;
   //aflam pozitia caracterului NULL(de cod ASCII 0)
   while (cuv[lung]!=0) lung++;
   cout<<"Lungimea sirului este= "<<lung<<endl;
   cout<<"Literele mici din cuvant sunt: "<<endl;
   for (i=0;i<=lung-1;i++)
      if (cuv[i]>='a' && cuv[i]<='z') cout<<cuv[i];
      //literele ocupa pozitii succesive in tabela ASCII
   return 0;
}
```

Problema 3. Același enunț ca la Problema 2.

Rezolvare cu șir propriu-zis:

```
#include <iostream>
#include <string.h>
// header-ul <string.h>  contine functii pentru siruri de caractere
using namespace std;
int main()
{   char cuv[20];
   int i,lung;
   cout << "Tastati cuvantul: "; cin>>cuv;
   lung=strlen(cuv);
   // functia strlen intoarce lungimea unui sir
   // functia strlen face parte din header-ul <string.h>
   cout<<"Lungimea sirului este= "<<lung<<endl;
   cout<<"Literele mici din cuvant sunt: "<<endl;
   for (i=0;i<=lung-1;i++)
      if (cuv[i]>='a' && cuv[i]<='z') cout<<cuv[i];
      //literele ocupa pozitii succesive in tabela ASCII
   return 0;
}
```

Am inclus header-ul <string.h>, care conține funcții de lucru cu șiruri de caractere.

Folosim funcția strlen() care calculează lungimea unui șir.

Deci, șirul de caractere este de fapt un tablou de caractere, care are ca ultim element un terminator de șir, numit caracterul NULL.

Problema 4. Citim de la tastatură două propoziții (șiruri de caractere formate din mai multe cuvinte). Fiecare propoziție se termină prin apăsarea tastei Enter. Să se afișeze:

- lungimea primei propoziții și literele mari ce o compun,
- toate vocalele ce compun a doua propoziție, precum și câte sunt.

Exemplu: Dacă vom tasta propozițiile „BuNa ZiUa !” și „Am onOarea Sa Va sALuT !”, atunci se vor afișa:

11 BNZU

AoOaeaaaAu 10

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
   char s1[50],s2[50];
   int l1, i,nrvocale=0;
   cout << "Prima propozitie este: "; cin.get(s1,50);
   cin.get();
   cout << "A doua propozitie este: "; cin.get(s2,50);
   cin.get();
   l1=strlen(s1); cout<<l1<<"  ";
   for (int i=0;i<=l1-1;i++)  // parcurgem prima propozitie
      if(s1[i]>='A' && s1[i]<='Z') cout<<s1[i];
      // literele mari ocupa pozitii consecutive in tabela ASCII
   cout<<'\n';
   //folosim caracterul new line(cod ASCII 10); are acelasi efect cu endl
   for (int i=0;i<=strlen(s2)-1;i++) //parcurgem a doua propozitie
      if   (s2[i]=='a'|| s2[i]=='e'|| s2[i]=='i'|| s2[i]=='o'|| s2[i]=='u'
        || s2[i]=='A'|| s2[i]=='E'|| s2[i]=='I'|| s2[i]=='O'|| s2[i]=='U')
        // testam toate vocalele mici sau mari
        {cout<<s2[i];nrvocale++;}
   cout<<"  "<<nrvocale;
   return 0;
}
```

Problema 5. Să se afișeze toate pozițiile unui caracter într-un șir dat. Pentru șirul „Ana si Mara fac case” și caracterul ‘a’ se vor afișa pozițiile 2, 8, 10, 13, 17.

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
```

```
{char a[200],*d,c;
 // *d va contine o adresa dintr-un sir
 cout<<"Tastati Sirul: ";cin.get(a,200);
 cout<<"Dati Caracterul: ";cin>>c;
 cout<<"Caracterul "<<c<<" apare in pozitiile: "<<endl;
 //pozitiile dintr-un sir incep de la 0(zero)
 d=strchr(a,c);
 while (d!=0)
   {cout<<"Pozitia "<<d-a<<endl;
            d++;
            d=strchr(d,c);}
 return 0;
}
```

Problema 6. Se citește de la tastatură un șir de caractere. Eliminați din el toate cifrele. Ce lungime are șirul obținut? Convertiți literele mici la litere mari în tot șirul obținut.

Exemplu: din șirul „Astazi 23martie 2015, la ora 14.59 citim versuri!” se obține șirul „Astazi martie, la ora . citim versuri!” și șirul „ASTAZI MARTIE, LA ORA . CITIM VERSURI!”.

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
   char sir1[100],sir2[100]; int i,j;
   cout << "Dati sirul initial: ";cin.get(sir1,100);
   j=-1; //indice pentru pozitii din sir2
   for(i=0;i<=strlen(sir1)-1;i++)
      if (sir1[i]<'0' || sir1[i]>'9') {j++;sir2[j]=sir1[i];}
      // depun in sir2 orice caracter care nu este cifra
   sir2[++j]=0;  //punem marcajul de sfarsit de sir
   cout<<"Sirul fara cifre este: "<<sir2<<endl;
   strupr(sir2);  //transformam lieterele mici in mari
   cout<<"Sirul cu litere mari este: "<<sir2;
   return 0;
}
```

Problema 7. Fie un șir de caractere. Să se inverseze ordinea cifrelor din șir.

Exemplu: dacă șirul inițial este „Elevii de la 8B au 347 note la 15 materii.”, atunci se obține șirul „Elevii de la 5B au 174 note la 38 materii.”.

```
#include <iostream>
#include <string.h>
```

```
using namespace std;

int main()
{
   char sir[100],c; int i,j;
   cout << "Tastati sirul: "; cin.get(sir,100);
   i=0; j=strlen(sir)-1;
   //Mergem cu 2 indici de la capetele sirului spre interior...
   while (i<j)  //..cat timp cei doi indici nu se depasesc
      {
         while (sir[i]<'0' || sir[i]>'9') i++;
         // cautam cifra cea mai din stanga inca neinversata
         while (sir[j]<'0' || sir[j]>'9') j--;
         // cautam cifra cea mai din dreapta inca neinversata
         if (i<j) {c=sir[i];sir[i]=sir[j];sir[j]=c;}
         // inversam cifrele sir[i] cu sir[j]
         i++;j--; // trecem peste cifrele tocmai inversate
      }
    cout<<"Sirul cu cifrele inversate este: "<<sir;
   return 0;
}
```

Problema 8. Din fișierul „sir.dat” se citește un șir de caractere. Să se scrie în fișierul „rezult.txt”, pe aceeași o linie, cel mai din stânga caracter din șir care nu se repetă și cel mai din dreapta caracter din șir care se repetă în șir; după fiecare caracter se scrie și de câte ori se repetă. Dacă nu există caractere care să îndeplinească condițiile de mai sus, atunci se scrie un mesaj corespunzător în fișierul „rezult.txt”.

Exemple:

- dacă fișierul „sir.dat” conține „caracatita si calamar”, atunci fișierul „rezult.txt” va conține s 1 r 2, pentru că cel mai din stânga caracter este ‘s’ cu o apariție și cel mai din dreapta caracter este ‘r’ cu 2 apariții;
- dacă fișierul „sir.dat” conține „abracadabra”, atunci fișierul „rezult.txt” va conține c 1 a 5, pentru că cel mai din stânga caracter este ‘c’ cu o apariție și cel mai din dreapta caracter este ‘a’ cu 5 apariții;
- dacă fișierul „sir.dat” conține „aaaabbbbb”, atunci fișierul „rezult.txt” va conține: „sirul nu are caractere care nu se repeta b 5“;
- dacă fișierul „sir.dat” conține „jkl347mno”, atunci fișierul „rezult.txt” va conține: „j 1 sirul nu are caractere care se repeta”.

```
#include <iostream>
#include <fstream>
#include <string.h>

using namespace std;
```

```
int mark[256];
//vector initializat la 0(zero),de dimensiunea tabelei ASCII
int main()
{
   char sir[150]; int i,lung,j,ok;
   ifstream f("sir.dat"); ofstream g("rezult.txt");
   f.get(sir,150); //citim un sir din fisier
   f.close();
   lung=strlen(sir);
   for (i=0;i<=lung-1;i++)
      mark[sir[i]]++; //marchez in vectorul mark numarul de aparitii al fiecarui caracter

   //caut cel mai din stanga caracter care NU se repeta
   //adica apare o singura data
   ok=0; //presupunem ca nu sunt caractere care nu se repeta
   for (j=0;j<=lung-1 && !ok;j++)
      if (mark[sir[j]]==1) {g<<sir[j]<<" "<< mark[sir[j]]<<"  ";ok=1;}
   if (ok==0) g<<"sirul nu are caractere care nu se repeta  ";

   //caut cel mai din dreapta caracter care se repeta
   //adica apare cel putin de 2 ori
   ok=0; //presupun ca nu sunt caractere care se repeta
   for (j=lung-1;j>=0 && !ok;j--)
      if (mark[sir[j]]>=2) {g<<sir[j]<<" "<< mark[sir[j]]<<endl;ok=1;}
   if (ok==0) g<<"sirul nu are caractere care se repeta   "<<endl;
   g.close();
   return 0;
}
```

Am folosit vectorul de marcaj mark[256] de dimensiunea tabelei ASCII, inițializat cu zero(pentru că l-am declarat înaintea tuturor funcțiilor din program). Pentru fiecare caracter din șir, am marcat în mark numărul de apariții al acelui caracter.

Apoi am parcurs șirul de la stânga la dreapta și am căutat primul caracter care apare doar o dată.

Am parcurs șirul de la dreapta la stânga și am căutat caracterul care apare de cel puțin 2 ori.

Probleme propuse

1. * Să se verifice dacă un cuvânt citit de la tastatură este palindrom. Exemplu: cuvântul „cojoc” este palindrom.
Indicație: se parcurge șirul cu doi indici, simultan din stânga și dreapta; se compară cele două carcatere indicate de indici.

2. * Să se determine de câte ori se găsește un cuvânt dat într-un text cunoscut.
Indicație: se folosește funcția strstr() pentru a căuta cuvântul în text, după care se șterge cuvântul din text și se reia căutarea.

3. Se citește un text de la tastatură astfel încât cuvintele să fie separate printr-un singur spațiu și imediat după ultimul cuvânt se pune punct. Câte cuvinte conține textul? Exemplu: „Cobori în jos Luceafăr blând." conține 5 cuvinte.

4. Realizați codificarea păsărească a unui cuvânt dat. Regula este: după fiecare vocală, se pune litera p urmată de acea vocală.
Exemplu: informatica devine ipinfopormapatipicapa

5. ** Citim un șir de la tastatură. Să se elimine caracterele duplicate din șir.
Exemplu: șirul „Peloponez din piatra" devine „lnz itr".
Indicație: folosim un vector de marcaj mark[256] în care marcăm numărul de apariții al fiecărui caracter în șir. Se copiază în alt șir toate caracterele care apar o singură dată.

6. ** Dintr-un fișier se citesc mai multe șiruri de caractere de pe linii diferite, formate din cuvinte separate de câte un spațiu. De la tastatură se citește un cuvânt. Afișati pe monitor cuvintele din fișier mai mari în sens lexicografic decât cuvântul citit de la tastatură.
Indicație: Din șirurile din fișier se separă cuvintele. Fiecare cuvânt se compară (folosind funcția strcmp()) cu cuvântul citit de la tastatură.

7. ** Comprimați un text citit de la tastatură după regula explicată în exemplele următoare.
Exemple:
- textul „ddddeeeazzzzzii" comprimat este „4d3e1a5z2i";
- textul „CoooLLOOsssUUl diiin RhhhoooDDDoos" comprimat este „1C3o2L2O3s2U1l1 1d3i1n3 1R3h3o3D2o1s"

8. *** Se citesc n cuvinte. Să se afișeze perechile de două cuvinte care rimează (au ultimele 2-3 caractere identice).
Indicație: memorăm cuvintele într-o matrice de caractere char matr[20][50], în care fiecare linie reprezintă un șir de caractere. Pentru fiecare pereche de două linii verificăm ultimele 2-3 caractere.

9. *** Citim un șir de forma „f=x+y-z" , unde x, y și z sunt numere reale. Afișați valoarea lui f. Exemplu: dacă citim „f=32+54.8-21.3", se va afișa „f=65.5".
Indicație: Identificăm pozițiile caracterelor =, + și -. Apoi se folosește funcția atof() și se obțin numerele reale din șir, cu care se fac calculele.

10. Dintr-un fișier se citesc numele a n personae formate numai din litere mici ale alfabetului englez. Fiecare nume se află pe câte o linie a fișierului și este format din numele de familie și 2-3 prenume, separate prin câte un spațiu (blank). Să se afișeze toate numele încât să fie scrise astfel: prima literă să fie mare la numele de familie și la cele 2-3 prenume, restul litere mici.

11. * Dintr-un fișier se citește un text. Textul conține cuvinte separate printr-un spațiu sau mai multe. Textul inițial, având spațiile de prisos eliminate (între cuvinte va rămâne numai câte un spațiu), se va scrie în alt fișier.
Indicație: Se citește textul caracter cu caracter din fișierul de intrare și nu se scrie decât câte un spațiu și toate celelalte caractere.

12. * Se citește un text format numai din litere și spații. Să se cripteze textul prin înlocuirea fiecărui caracter cu caracterul aflat peste k poziții în tabela ASCII. Valoarea numerică k se citește de la tastatură.

13. ** Considerăm că două cuvinte rimează dacă sufixele începând de la ultima vocală sunt identice. Scrieți un program care citește două cuvinte a și b și verifică dacă ele rimează.
Exemplu: cuvintele raza și lumineaza rimează pentru că au aza ca parte comună.
Indicație: se compară caracterele celor două cuvinte de la dreapta la stânga. Comparările se opresc când se găsesc două caractere diferite. Dacă caracterele imdiat precedente sunt vocale identice, atunci cuvintele rimează.

14. Se citește o frază. Să se șteargă consoanele din ea.
Indicație: se consideră consoană un caracter care este literă, dar nu este vocală.

15. * În fișierul cuvinte.dat se află mai multe cuvinte separate prin câte un spațiu. Să se scrie un program care citește cuvintele din fișier și le scrie în fișierul cuvinte.txt în ordine alfabetică.
Exemplu:
cuvinte.dat conține: gigi mara val casa
cuvinte.txt va conține: casa gigi mara val
Indicație: se depun cuvintele într-o matrice char mat[30][50], câte unul pe fiecare linie. Folosind o metodă de sortare cunoscută, se ordonează liniile matricei. **Atenție:** compararea a două șiruri de caractere se face cu funcția strcmp(), iar copierea unui șir în altul se face cu strcpy().

16. *** Să se afișeze toate prefixele și sufixele unui cuvânt citit de la tastatură. Prefixele unui cuvânt sunt compuse din minim un caracter și maxim toate caracterele, citite de la stânga la dreapta. Sufixele unui cuvânt sunt compuse din minim un caracter și maxim toate caracterele, generate dinspre dreapta spre stânga.

Exemplu: cuvântul calaret are prefixele: c ca cal cala calar calare calaret, iar sufixele sunt: t et ret aret laret alaret calaret

17. Scrieţi un program C/C++ care citeşte de la tastatură un text şi afişează pe ecran, pe o linie și separate de câte un spațiu, toate secvenţele formate din câte două caractere identice. Exemplu: dacă textul citit este „Copiii studiaza Zoologia”, atunci se vor afișa secvențele: ii ii oo.

Capitolul 4.

Tipul înregistrare (struct)

Noțiuni teoretice

V-ați gândit cum vom proceda dacă dorim să memorăm și să prelucrăm mai multe date pentru același obiect?

De exemplu, dorim să reținem pentru n elevi următoarele informații: numele, prenumele și vârsta. Utilizând noțiunile învățate în capitolele anterioare, într-un astfel de caz putem să utilizăm tablouri unidimensionale. Pentru datele celor n elevi trebuie să folosim 3 tablouri unidimensionale (vectori): un vector care să rețină numele, un vector care să rețină prenumele și un vector care să rețină vârsta. O astfel de metodă este destul de anevoioasă, mai ales dacă volumul informațiilor pe care vrem să le reținem crește.

Pentru astfel de situații o metodă mult mai simplă și eficientă este utilizarea tipului înregistrare – STRUCT – un tip de date definit de către programator.

Forma generală a tipului de date STRUCT este:

```
struct nume
{
tip nume_variabile;
tip nume_variabile;
.....
} lista_variabile;
```

Pentru exemplul de mai sus vom defini următorul tip de date:

```
struct elev
{
char nume[20],prenume[20];
int varsta;
}elev1,elev2;
```

elev – este numele tipului de date

nume[20] – reprezintă câmpul care va reține numele elevului (este de tipul char)

prenume[20] – reprezintă câmpul care va reține prenumele elevului (este de tipul char)

varsta – reprezintă câmpul care va reține vârsta elevului (este de tipul int)
elev1 și **elev2** sunt 2 variabile de tipul **elev**

Definirea tipului de date se poate face în interiorul funcției main() sau înaintea funcției main() (caz pe care îl recomandăm).

Referirea la câmpurile unei variabile de acest tip se face cu ajutorul operatorului de selecție „ **.** ” (punct) astfel: **nume_variabilă.nume_câmp**

Exemple:
elev1.nume – ne referim la numele elevului
elev1.varsta – ne referim la vârsta elevului

Probleme rezolvate

Problema 1. Să se scrie un program care citește de la tastatură numele, prenumele și vârsta unui elev și afișează pe ecran informațiile citite.

```
#include <iostream>
#include <string.h>
using namespace std;

struct elev  //definirea tipului de date elev
     {
     char nume[20],prenume[20];
     int varsta;
     };

int main()
{   elev e; //declararea variabilei de tip elev
   cout<<"Introduceti numele elevului: "; //se cere introducerea numelui
   cin>>e.nume;
   cout<<"Introduceti prenumele elevului: "; //se cere introducerea prenumelui
   cin>>e.prenume;
   cout<<"Introduceti varsta elevului: "; //se cere introducerea varstei
   cin>>e.varsta;
      //afisarea datelor citite
   cout<<e.nume<<" "<<e.prenume<<" "<<e.varsta;
   return 0; }
```

Problema 2. Să se scrie un program care calculează suma a două fracții utilizând tipul înregistrare.

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

```
struct fractie
     {
     int numarator,numitor;
     };

int main()
{
   fractie f1,f2,f;
   cout<<"Introduceti numaratorul pentru prima fractie: ";cin>>f1.numarator;
   cout<<"Introduceti numitorul pentru prima fractie: ";cin>>f1.numitor;
   cout<<"Introduceti numaratorul pentru a doua fractie: ";cin>>f2.numarator;
   cout<<"Introduceti numitorul pentru a doua fractie: ";cin>>f2.numitor;
   f.numitor=f1.numitor*f2.numitor;
   f.numarator=f1.numarator*f2.numitor+f2.numarator*f1.numitor;
   cout<<f1.numarator<<"/"<<f1.numitor<<"+"<<f2.numarator<<"/"<<f2.numitor<<"=";
   cout<<f.numarator<<"/"<<f.numitor;

   return 0;
}
```

Problema 3. Să se scrie un program care citește de la tastatură numele, prenumele, media la matematică și media la limba română și afișează pe ecran disciplina la care elevul este corigent. Dacă nu este corigent la nici o disciplină se va afișa un mesaj corespunzător.

```
#include <iostream>
#include <string.h>
using namespace std;

struct elev
     {
     char nume[20],prenume[20];
     float mmate,mrom;
     };

int main()
{
   elev e;
   cout<<"Introduceti numele elevului: ";cin>>e.nume;
   cout<<"Introduceti prenumele elevului: ";cin>>e.prenume;
   cout<<"Introduceti media la matematica: ";cin>>e.mmate;
   cout<<"Introduceti media la limba romana: ";cin>>e.mrom;

   if (e.mmate<5)
      cout<<"Elevul "<<e.nume<<" "<<e.prenume<<" este corigent la matematica."<<endl;
   if (e.mrom<5)
      cout<<"Elevul "<<e.nume<<" "<<e.prenume<<" este corigent la limba romana."<<endl;
```

```
   if ((e.mmate>=5) && (e.mrom>=5))
      cout<<"Elevul "<<e.nume<<" "<<e.prenume<<" nu este corigent la nicio materie.";

   return 0;
}
```

Problema 4. La un concurs de Informatică participă n elevi. Pentru fiecare elev se cunoaște: numele, prenumele, punctajul la proba 1, punctajul la proba 2. Să se scrie un program care afișează pentru fiecare elev punctajul total obținut la concurs.

```
#include <iostream>
#include <string.h>
using namespace std;

struct elev
     {
     char nume[20],prenume[20];
     int p1,p2;
     };

int main()
{
   int n,i;
   elev e[20];

   cout<<"Introduceti numarul de elevi: ";cin>>n;
   for (i=1;i<=n;i++)
   {cout<<"Introduceti numele elevului: ";cin>>e[i].nume;
   cout<<"Introduceti prenumele elevului: ";cin>>e[i].prenume;
   cout<<"Introduceti punctajul la proba 1: ";cin>>e[i].p1;
   cout<<"Introduceti punctajul la proba 2: ";cin>>e[i].p2;}

   for (i=1;i<=n;i++)
   cout<<e[i].nume<<" "<<e[i].prenume<<" a obtinut "<<e[i].p1+e[i].p2<<" puncte."<<endl;

   return 0; }
```

Problema 5. Un vector de n înregistrări conține informațiile despre produsele dintr-un magazin: numele produsului, prețul și cantitatea. Să se scrie un program care afișează prețurile produselor după o scumpire cu k la sută, unde k este un număr natural care se citește de la tastatură.

```
#include <iostream>
```

```
#include <string.h>
using namespace std;

struct produse
     {
     char denumire[20];
     int pret, cantitate;
     };

int main()
{
   int n,i,k;
   produse v[20];

   cout<<"Introduceti numarul de produse: ";cin>>n;
   for (i=1;i<=n;i++)
   {cout<<"Denumirea produsului "<<i<<": ";cin>>v[i].denumire;
   cout<<"Pretul produsului: ";cin>>v[i].pret;
   cout<<"Cantitatea produsului: ";cin>>v[i].cantitate;}

   cout<<"Introduceti valoarea procentului cu care se scumpesc produsele: ";cin>>k;

   cout<<"Pretul produselor dupa scumpire: "<<endl;
   for (i=1;i<=n;i++)
   cout<<v[i].denumire<<" --> "<<v[i].pret+(float)(v[i].pret*k)/100<<endl;

   return 0;
}
```

Problema 6. Se citesc de la tastatură n perechi de numere naturale de forma (a,b). Să se scrie un program care afișează perechile formate din numere pare.

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

struct perechi
     {
     int a;
     int b;
     };

int main()
{
   int n,i;
   perechi v[20];

   cout<<"Introduceti numarul de perechi: ";cin>>n;
   for (i=1;i<=n;i++)
```

```
    {cout<<"Perechea "<<i<<": "<<endl;
    cout<<"a=";cin>>v[i].a;
    cout<<"b=";cin>>v[i].b;}

    cout<<"Perechile formate din numere pare sunt: "<<endl;
    for (i=1;i<=n;i++)
        if ((v[i].a%2==0) && (v[i].b%2==0))
            cout<<"("<<v[i].a<<","<<v[i].b<<")"<<endl;

    return 0;
}
```

Probleme propuse

1. Pentru produsele dintr-un magazin se cunosc următoarele informații: denumirea produsului, prețul, cantitatea (numărul de exemplare prezente în magazin). Să se scrie un program care afișează valoarea produselor din magazin.

2. Să se scrie un program care calculează suma a 3 fracții. Modificați programul astfel încât fracția rezultată să fie ireductibilă.
Indicație: numărătorul și numitorul se simplifică cu cmmdc-ul lor.

3. * La concursul „Micul Savant” au participat n elevi. Fiecare elev a obținut un punctaj cuprins între 0-100. Să se scrie un program care afișează elevii care au urcat pe podium.

4. * Din fișierul triunghiuri.in se citesc lungimile laturilor pentru n triunghiuri. Să se scrie un program care afișează numărul triunghiurilor isoscele, numărul triunghiurilor echilaterale și numărul triunghiurilor oarecare. Fișierul triunghiuri.in are următoarea structură: pe prima linie se află un număr natural n care reprezintă numărul de triunghiuri iar pe următoarele n linii se află câte 3 numere naturale care reprezintă dimensiunile laturilor pentru fiecare triunghi.

5. Într-o clasă sunt n elevi. Fiecare elev are, pe semestrul I, la matematică 4 note și nota de la teză. Să se scrie un program care afișează numele elevului care a obținut cea mai mare medie pe semestrul I.

6. Se citesc de la tastatură n fracții de forma a/b unde a și b sunt două numere naturale. Să se scrie un program care afișează:
- fracțiile supraunitare;
- numărul fracțiilor echiunitare.

7. ** În fișierul note.in se află notele obținute de către elevii unei clase la disciplina Informatică. Fișierul are următoarea structură:
- pe prima linie numărul de elevi n;
- pe următoarele n linii, separate cu câte un spațiu, informațiile corespunzătoare fiecărui elev: 4 note și numele elevului.
Să se scrie un program care afișează în fișierul note.out următoarele:
- pe prima linie, numărul de elevi corigenți;
- pe următoarele linii, câte un elev pe linie, elevii care au media la disciplina Informatică mai mare decât media clasei.

8. ** Se citesc de la tastatură n perechi de numere naturale de forma (a,b). Să se scrie un program care afișează numărul perechilor formate din numere prime și perechile care au media aritmetică în intervalul [x,y], unde x și y sunt două numere naturale citite de la tastatură.

9. *** Fișierul populatie.in conține evidența populației din n orașe astfel:
- pe prima linie se află un număr natural n, reprezentând numărul orașelor;
- pe următoarele n linii se află evidența populației din cele n orașe în următoarea ordine: codul județului (format din 2 caractere), numele orașului (maxim 20 de caractere) și numărul de locuitori.
Să se scrie un program care afișează numărul de locuitori din fiecare județ în parte.

10. * La un concurs sportiv – faza județeană – s-au prezentat n concurenți și fiecare concurent a susținut câte 3 probe, la fiecare probă obținând un punctaj cuprins între 0 și 100. Punctajul final se obține din însumarea punctajelor de la cele 3 probe. La faza națională a concursului s-au calificat concurenții care au obținut minim 200 de puncte. Să se scrie un program care afișează numărul și numele concurenților care vor participa la faza națională. Dacă nu există concurenți participanți la faza națională programul va afișa mesajul: "Nici un concurent nu s-a calificat la faza națională."

11. ** De la tastatură se citesc n perechi de numere naturale de forma (a,b,c). Să se scrie un program care afișează pe ecran perechile al căror elemente au cel mai mare divizor comun un număr multiplu de k, unde k este un număr natural care se citește de la tastatură.

12. *** Evidența absențelor a n elevi dintr-o clasă se centralizează cu ajutorul unui tip înregistrare. Pentru fiecare elev se cunoaște: numele și prenumele, numărul de absențe (motivate și nemotivate) pe semestrul I, numărul de absențe motivate pe semestrul I, numărul de absențe (motivate și nemotivate) pe semestrul al II-lea, numărul de absențe motivate pe semestrul al II-lea. Pe fiecare semestru, la 10 absențe nemotivate, elevului i se scade 1 punct la purtare. Să se scrie un program care afișează pe ecran pentru fiecare elev media anuală la purtare.

13. ** Se citesc de la tastatură numitorii și numărătorii a două fracții. Să se scrie un program care simplifică suma și produsul celor două fracții și afișează pe ecran numitorul și numărătorul fracțiilor rezultate.

14. ** Pentru evidența produselor dintr-un magazin, se definește o structură cu următoarele informații: codul, denumirea, prețul produsului. Citiți informațiile a n produse și afișați informațiile despre produse după o schimbare a prețurilor care se va realiza astfel: valoarea prețului va fi k dacă codul produsului este un număr par și valoarea prețului va fi r dacă codul produsului este în intervalul [a,b]. Numerele k, r, a, b sunt numere naturale care se citesc de la tastatură.

15. ** Citim de la tastatură un șir de forma:
produs1=cantitate1; produs2=cantitate2; ….; produsn=cantitaten
Produsele și cantitățile se pot repeta. Ele reprezintă produsele lanțurilor de magazine Kaufland din țară. Numele produselor conțin numai litere mici ale alfabetului englez, iar cantitățile sunt numere naturale.
Să se afișeze pentru fiecare produs, ce cantitate se află în tot lanțul de magazine.
Exemplu: Dacă șirul citit este de forma:
zahar=34;paine=1023;bere=234;paine=344;mere=345;zahar=55;bere=206
atunci se vor afișa:
zahar=79;paine=1367;bere=440;mere=345
Indicație: folosim un vector de înregistrări (struct) în care plasăm fiecare produs doar o dată (la momentul primei apariții în șir); de câte ori întâlnim în șir un același produs, îi adună cantitatea; în final afișăm produsele și cantitățile totale.

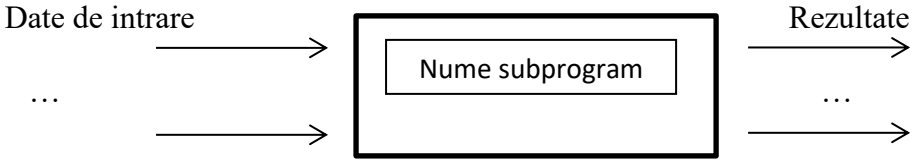
Capitolul 5.

Subprograme definite de utilizator

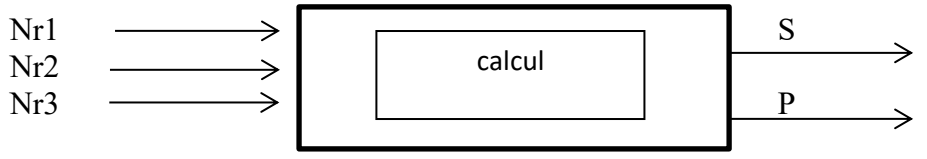
Noțiuni teoretice

Subprogramul este un set de instrucțiuni care rezolvă o sarcină bine definită. El primește una sau mai multe date la intrare și poate returna unul sau mai multe rezultate(date de ieșire).

Aspectul general al unui subprogram este cel din figura următoare:



Exemplu: un subprogram numit calcul, care calculează suma S și produsul P a trei numere reale Nr1, Nr2, Nr3.



Descrierea în limbajul pseudocod:

```
Subprogram calcul (reale NR1, NR2, NR3, S, P)
S= NR1+NR2+NR3
P=NR1*NR2*NR3
Sfârșit_Subprogram
```

Descrierea subprogramului în limbajul C++ este următoarea:

```
void calcul (float NR1, float NR2, float NR3, float &S, float &P)
{
S=NR1+NR2+NR3;
P=NR1*NR2*NR3;
```

```
}
```

Parametrii folosiți la descrierea subprogramului se numesc **parametrii formali**.

Parametrii formali de intrare sunt NR1, NR2, NR3.

Parametrii formali de ieșire sunt S, P (ei sunt precedați de semnul & pentru că în ei se obțin rezultatele subprogramului).

Cuvântul cheie void indică că în numele funcției nu se întoarce nimic (nicio valoare) după apel.

Subprogramele se apelează (se folosesc) în programe, pentru a rezolva diferite probleme.

Exemplu de problemă: Se citesc 6 numere reale de la tastatură; să se calculeze și să se afișeze suma și produsul celor 6 numere; se va folosi subprogramul calcul, definit anterior.

Programul (care folosește/apelează subprogramul calcul) în pseudocod este următorul:

```
Program
Reale a,b,c,d,e,f,s1,p1,s2,p2,S,P
Citește a,b, c, d, e, f
Calcul(a,b,c,s1,p1)
Calcul(d,e,f,s2,p2)
S=s1+s2
P=p1*p2
Afișează S, P
Sfârșit_Program
```

Programul principal, care apelează/folosește subprogramul calcul în limbajul C++, este următorul:

```
int main()
{
float a,b,c,d,e,f,s1,p1,s2,p2,S,P;
cin>>a>>b>>c>>d>>e>>f;
calcul(a,b,c,s1,p1); // apel subprogram
calcul(d,e,f,s2,p2); // apel subprogram
S=s1+s2;
P=p1*p2;
cout<<S<<" "<<P;
return 0;}
```

Observăm că subprogramul calcul este apelat de două ori; prima dată calculează suma și produsul primelor trei numere în variabilele s1 și p1, apoi calculează suma și produsul următoarelor trei numere în variabilele s2 și p2. Rezultatele finale S și P se obțin prin folosirea rezultatelor intermediare s1, p1, s2, p2.

Parametrii folosiți la apelarea subprogramului se numesc **parametrii efectivi**.

Parametrii efectivi transmiși prin valoare sunt NR1, NR2, NR3.

Parametrii efectivi transmiși prin referință sunt S, P (în ei se obțin rezultatele).

În limbajul C++, subprogramele se numesc **funcții**. O funcție poate întoarce în numele ei un rezultat. De exemplu, funcția calcul poate întoarce în numele ei suma, iar produsul în parametrul P. În această situație, programul sursă complet scris în limbajul C++ standard, este următorul:

```
#include <iostream>
using namespace std;
float calcul (float NR1, float NR2, float NR3, float &P)
        {
        P=NR1*NR2*NR3;
        //depune produsul in parametrul formal P transmis prin referinta &
        float S=NR1+NR2+NR3;
        return S; //întoarce suma  în numele funcției calcul
        }
int main()
        {
        float a,b,c,d,e,f,s1,p1,s2,p2,S,P;
        cin>>a>>b>>c>>d>>e>>f;
        s1=calcul(a,b,c,p1);
        s2=calcul(d,e,f,p2);
        S=s1+s2;
        P=p1*p2;
        cout<<S<<" "<<P;
        return 0;
        }
```

Probleme rezolvate

Problema1. Descrieți un subprogram numit medaritm, care primește la intrare trei numere naturale și calculează media lor aritmetică. Descrieți un subprogram numit medgeom, care primește la intrare trei numere naturale și calculează media lor geometrică; folosiți aceste subprograme pentru a calcula media aritmetică și geometrică a trei numere naturale citite de la tastatură.

Programul complet reprezentat în pseudocod este format din cele două subprograme și din programul principal.

```
Subprogram medaritm(naturale a,b,c; real meda)
    meda=(a+b+c)/2
```

```
Sfârşit_Subprogram
Subprogram medgeom(naturale a,b; real medg)
        P=a*b*c
        medg=radicalordin3(p)
Sfârşit_Subprogram

Program
        naturale x,y,z; reale ma,mg
         citeste x,y,z
        medaritm(x,y,z,ma) //calculeaza media aritmetica dintre x, y și z
        medgeom(x,y,z,mg)  //calculează media geometrică dintre x,y,z
        afiseaza ma, mg
Sfârșit_Program
```

Programul sursă complet scris în limbajul C++ standard este următorul:

```
#include <iostream>
#include <math.h>
using namespace std;
void medaritm(int a, int b, int c, float &meda)
        {
        meda=(a+b+c)/3.0;
        }
void medgeom(int a, int b, int c, float &medg)
        {
        float p=a*b*c;
        // media geometrica a 3 numere este radical de ordin 3 din produsul lor
        // radical de ordin trei din p  se poate scrie ca p la puterea 1/3
        // iar p la 1/3 se calculeaz cu functia matematica power(p,1/3)
        medg=pow(p,1.0/3);
        }
int main()
        {
        int x,y,z; float ma,mg;
        cout<<"Tastati cele trei numere=";
        cin>>x>>y>>z;
        medaritm(x,y,z,ma);
        medgeom(x,y,z,mg);
        cout<<ma<<" "<<mg;
        return 0;
        }
```

Problema 2. Scrieți descrierea (definirea) subprogramului arietri, care calculează aria unui triunghi oarecare pentru care se dau lungimile laturilor. Folosiți acest subprogram pentru a calcula și afișa ariile a n triunghiuri oarecare, ale căror laturi se citesc de la tastatură. Numărul n se dă de la tastatură.

Aspectul general al subprogramului arietri folosit:

Programul complet reprezentat în pseudocod:

```
Subprogram arietri(reale a,b,c,aria)
      p=(a+b+c)/2
      aria=radical(p*(p-a)*(p-b)*(p-c))  //formula lui Heron
Sfârşit_Subprogram

Program
      reale x,y,z, ar
      naturale i, n
      citeşte n
      Pentru i=1,n,1  execută
            citeşte x, y, z
            arietri(x,y,z,ar)
            afişează ar
      Sfârşit_Pentru
Sfârşit_Program
```

Programul sursă complet scris în limbajul C++ este următorul:

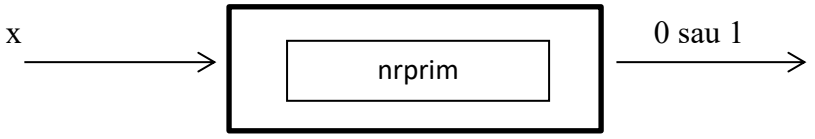
```
#include <iostream>
#include <math.h>
using namespace std;
float arietri(float a,float b,float c)
{
float p=(a+b+c)/2;
return  sqrt(p*(p-a)*(p-b)*(p-c));
//formula lui Heron
}
int main()
{
float x,y,z,ar;  int i, n;
cout<<"Dati numarul de triunghiuri: ";
cin>>n;
for (i=1; i<=n; i++)
      {
      cout<<"Dati laturile triunghiului "<<i<<" : ";
      cin>>x>>y>>z;
```

```
        ar=arietri(x,y,z);
        cout<<"Aria="<<ar<<endl;
        }
return 0;
}
```

Problema 3. De pe prima linie a fişierului numere.txt se citeşte numărul natural n, iar de pe următoarea linie se citesc n numere naturale (care sunt separate prin spaţii). Să se afişeze primul şi ultimul număr prim din fişier.

Exemplu: dacă fișierul conține următoarele 11 numere 56 85 2 17 88 63 1 0 54 13 15, atunci se vor afișa 2 și 13.

Folosim un subprogram care verifică dacă un număr natural este prim sau nu. Aspectul general al subprogramului nrprim folosit este următorul:



Programul sursă complet scris în limbajul C++ este următorul:

```
#include <iostream>
#include <math.h>
#include <fstream>
using namespace std;
int nrprim (int x)
        {
        if ((x==0) ||(x==1))  return 0; // 0 și 1 nu sunt numere prime
        if (x==2) return 1 ; // 2 este număr prim
        if (x%2==0)  return 0; // orice număr care se divide cu 2 nu este prim
        // verificăm dacă x se divide cu vreun număr impar dintre 3 și radical din x
        for(int i=3;i<=sqrt(x); i=i+2)
                if (x%i==0) return 0;  // dacă se divide cu i, atunci x nu este număr prim
        return 1;  // dacă nu s-a divizat cu nimic, atunci x este număr prim
        // folosim instrucțiunea return pentru a părăsi forțat și imediat subprogramul
        // valorile întoarse în numele funcției: 1 indică x număr prim; 0 indică x neprim
        }
int main()
        {
        int n, i, j, x, primul, ultimul;
        ifstream f("numere.txt");
        f>>n;
        for(i=1;i<=n ;i++)  //caut primul număr prim
                {f>>x;
                if (nrprim(x)==1) {primul=x; break;}
                // instructiunea break forteaza parasirea structurii curente
```

```
            }
    for(j=i+1;j<=n;j++)  //parcurge si restul numerelor din fisier
            {f>>x;
            if (nrprim(x)==1) ultimul=x;
            }
    cout<<primul<<" "<<ultimul;
    f.close();
    return 0;
    }
```

Problema 4. Scrieți un program în limbajul C++ standard care să calculeze și să afișeze cel mai mare divizor comun dintre patru numere naturale citite de la tastatură. Se va folosi o funcție care calculează și întoarce în numele ei cel mai mare divizor comun (cmmdc) dintre două numere naturale.

```
#include <iostream>
using namespace std;
int cmmdc(int a, int b)  // calculează cel mai mare divizor comun dintre numerele a și b
    {
    int rest;
    while (b!=0)
            {rest=a%b; a=b;b=rest;}
    return a; //intoarce valoarea din variabila a in numele functiei
    }
int main()
    {
    int x, y , z, t, d1, d2, d3;
    cout<<" Dati cele 4  numere: ";
    cin>>x>>y>>z>>t; //citim cele 4 numere
    d1=cmmdc(x,y); //calculăm cmmdc-ul primelor 2 numere
    d2=cmmdc(d1,z);  //calculăm cmmdc dintre d1 și al treilea număr
    d3=cmmdc(d2,t);  //calculăm cmmdc dintre d2 și al patrulea număr
    cout<<"cel mai Mare divizor comun al celor patru numere este="<<d3;
    return 0;
    }
```

Problema 5. Scrieți un program în limbajul C++ standard care calculează și afișează suma finală a 3 vectori; fiecare vector are dimensiunea m și este format din numere reale. Se va folosi funcția calcul care calculează suma a doi vectori; rezultatul funcției este un vector obținut prin adunarea element cu element a celor doi vectori. De asemenea se va afișa și vectorul intermediar obținut prin adunarea primilor 2 vectori; pentru afișarea unui vector se va crea funcția afis.

Valoarea lui m se citește de pe prima linie a fișierului vectori.txt, iar elementele celor 3 vectori se citesc de pe următoarele 3 linii ale fișierului.

Exemplu: dacă fișierul vectori.txt are următorul conținut:

5
3 6 2.2 8 4
7 3.4 6 5 8
9.2 2 3 5 1
atunci se vor afișa:
vectorul intermediar este: 10 9.4 8.2 13 12
vectorul suma final este: 19.2 11.4 11.2 18 13

```
#include <fstream>
using namespace std;
void calcul (int dim, float a[50], float b[50], float c[50])
	{
	for (int i=0; i<dim;i++)
		c[i]=a[i]+b[i];
	}
void afis(int dim, float v[50])
	{
	for (int i=0; i<dim;i++)	cout<<v[i]<<"  ";
	cout<<endl;
	}
int main()
	{
	int m; float v1[50], v2[50], v3[50], vi[50], vf[50];
	ifstream f("vectori.txt");
	f>>m;
	for (int i=0;i<m;i++) f>>v1[i]; //citește elementele primului vector
	for (int i=0;i<m;i++) f>>v2[i]; //citește elementele celui de-al doilea vector
	calcul(m, v1, v2, vi);  // calculează vectorul sumă vi dintre primii doi vectori
	cout<<"vectorul intermediar este: "; afis(m,vi);
	for (int i=0;i<m;i++) f>>v3[i]; //citește elementele celui de-al treilea vector
	calcul(m, vi, v3,vf); // calculează vectorul suma final vf dintre vi și vf
	cout<<"vectorul suma final este: "; afis(m,vf);
	f.close();
	return 0;
	}
```

Problema 6. Fie următorul șir de numere naturale: 1 1 2 3 5 8 13 21 34 55 89 144 …

Șirul se numește Fibonacci și se definește astfel: primul termen este f1=1; al doilea termen este f2=1; începând cu rangul 3, termenii se obțin prin suma celor 2 precedenți f3=f1+f2.

Se citește numărul natural n și apoi n ranguri(numere naturale). Să se afișeze termenii din șir de ranguri date. Se va folosi o funcție care generează termenii șirului până la rangul r dat ca parametru.

```
#include <iostream>
using namespace std;
unsigned long fib(int r)
        {
        if (r==1 || r==2) return 1;
        unsigned long f1, f2, f3;
        f1=f2=1;
        for (int i=3;i<=r;i++)
        {
        f3=f1+f2; f1=f2;f2=f3;
        }
        return f3;
        }
int main()
        {
        int n, r;
        cout << "Dati numarul de ranguri n="; cin>>n;
         for (int i=1;i<=n;i++)
                {
                cout <<" dati rangul r="; cin>>r;
                cout <<" Termenul Fibonnacci de rang "<<r<<" este:"<<fib(r)<<endl;
                }
        return 0;
        }
```

Temă de reflecție: concepeți un program care rezolvă mai rapid cerințele problemei.

Indicație: întâi sortați(ordonați) rangurile, apoi folosiți funcția fib pentru a genera termenii șirului.

Problema 7. Se citesc n numere naturale. Pentru fiecare număr să i se afișeze inversul. Folosiți o funcție care separă cifrele unui număr și-i construiește inversul. Exemple: dacă se dă numărul natural 7639, atunci inversul său este 9367; pentru numărul 85000346, inversul este 64300058.

```
#include <iostream>
using namespace std;
unsigned long invers(unsigned long nr)
{
      unsigned long x=0;
      while (nr!=0)
         {
     int cifra=nr%10; // separa cifra
```

```
    x=x*10+cifra; // adauga cifra la numarul invers
    nr=nr/10;  //renunta la cifra din numarul initial
     }
     return x; // intoarce inversul in numele functiei
}
int main()
{
   int n;
   unsigned long numar;
   cout << "Dati numarul de numere n="; cin>>n;
   for (int i=1;i<=n;i++)
         {
          cout<<" Dati un numar: "; cin>>numar;
          cout<<" Inversul este: "<<invers(numar)<<endl<<endl;
         }
   return 0;
}
```

Problema 8. Vectorul V se completează cu n numere naturale citite din fișierul „nr.txt”. Pe prima linie a fișierului se află n (numărul de elemente), iar pe a doua linie se găsesc numerele separate prin spații. Scriem un subprogram Bubble-Sort-optimizat care să ordoneze crescător un vector și un subprogram afis care să afișeze un vector. Folosim cele două subprograme pentru a afișa elementele lui V în ordine crescătoare, precum și evoluția vectorului V pe parcursul sortării.

Vom folosi cunoscuta și vechea sortare Bubble-Sort, dar o vom îmbunătăți și o vom face mai eficientă în timp și mai rapidă în execuție.

Programul scris în limbajul C++ este următorul:

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
int n, v[200];
void afis(int n, int v[200])
{
     int j;
     for (j=1;j<=n;j++) {cout<<setw(2)<<v[j]<<" "; }
     //functia setw se gaseste in biblioteca iomanip
     //functia setw(2) seteaza afisarea unei valori pe exact doua pozitii
     //pentru estetica afisarii
     cout<<endl;
}
void Bubble_Sort_optimizat(int n, int v[100])
{
     int i, poz,ok,aux, pozultim;
     poz=pozultim=n; //numarul total de elemenete din V
//1
```

```
do
 {ok=1; //presupunem vectorul V ordonat
  i=1;
  //parcurgem V pana la pozitia poz a ultimei interschimbari anterioare facute
  //si eventual facem interschimbarile necesare
  //2
  while(i<poz)
   {if(v[i]>v[i+1]) {aux=v[i]; v[i]=v[i+1];v[i+1]=aux;
                ok=0;pozultim=i;}  //memoram pozitia ultimei schimbari
    i++;
   }
   //2
   poz=pozultim; //actualizam pozitia pana la care va merge while
   afis(n,v);  //afisam vectorul V
 }
 while ((ok==0)&& poz>1); //se reia daca am facut interschimari
//1
  }
  int main()
        {
        ifstream f("nr.txt");
        f>>n; for(int i=1;i<=n;i++) f>>v[i];
        cout<<"Vectorul initial V este:"<<endl;
        afis(n,v);
        cout<<endl<<"Evolutia vectorului V este:"<<endl<<endl;
        Bubble_Sort_optimizat(n,v);
        return 0;
        }
```

Ideea care eficientizează Bubble-Sort-vechi se bazează pe observația următoare: la fiecare execuție a blocului //1 do{}while //1, cel puțin ultimele două elemente din V sunt ordonate.

Atunci, ideea este: la fiecare efectuare a unei interschimbări de două elemente, reținem poziția acestei interschimbări; în acest fel, variabila pozultim va reține poziția ultimei interschimbări efectuate; iar blocul //2 while(i<poz) //2 va parcurge vectorul V numai până la poziția ultimei interschimbări efectuate.

Bubble-Sort-optimizat obține o îmbunătățire de aproximativ 50% a vechiului algoritm Bubble-Sort-vechi !

În acest fel, Bubble-Sort-optimizat devine mai bună decât metodele: aflarea minimului/maximului (selecție), heap, numărare, inserție. Dar, rămâne mai slabă decât metodele: Shell_Sort (micșorarea incrementului), Merge_Sort (fuziune, interclasare), Quick_Sort (sortare rapidă).

Exemplu:

Fie vectorul V format din următoarele 22 numere:

5 9 4 3 6 11 2 23 24 8 26 26 7 26 26 28 32 29 29 10 29 29

Bubble-Sort-vechi parcurge vectorul V de 22 ori.
Bubble-Sort-optimizat parcurge vectorul V de numai 12 ori !!
Evoluția vectorului V în timpul sortării este următoarea:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	pozultim
5	4	3	6	9	2	11	23	8	24	26	7	26	26	26	28	29	29	10	29	29	32	21
4	3	5	6	2	9	11	8	23	24	7	26	26	26	26	28	29	10	29	29	29	32	18
3	4	5	2	6	9	8	11	23	7	24	26	26	26	26	28	10	29	29	29	29	32	17
3	4	2	5	6	8	9	11	7	23	24	26	26	26	26	10	28	29	29	29	29	32	16
3	2	4	5	6	8	9	7	11	23	24	26	26	26	10	26	28	29	29	29	29	32	15
2	3	4	5	6	8	7	9	11	23	24	26	26	10	26	26	28	29	29	29	29	32	14
2	3	4	5	6	7	8	9	11	23	24	26	10	26	26	26	28	29	29	29	29	32	13
2	3	4	5	6	7	8	9	11	23	24	10	26	26	26	26	28	29	29	29	29	32	12
2	3	4	5	6	7	8	9	11	23	10	24	26	26	26	26	28	29	29	29	29	32	11
2	3	4	5	6	7	8	9	11	10	23	24	26	26	26	26	28	29	29	29	29	32	10
2	3	4	5	6	7	8	9	10	11	23	24	26	26	26	26	28	29	29	29	29	32	9
2	3	4	5	6	7	8	9	10	11	23	24	26	26	26	26	28	29	29	29	29	32	9

Prima linie conține poziţiile curente din vectorul V.
Ultima coloană indică valoarea indicelui unde s-a realizat ultima interschimbare (pozultim).
Observăm cum pozultim face „salturi" peste întregi zone din vectorul V.

Probleme propuse

1. Se citesc de la tastatură n numere reale. Să se calculeze și afișeze suma lor, folosind apeluri ale unei funcții ce calculează suma a câte două numere reale.
Indicație: se calculează suma primelor două, rezultatul se sumează cu al treilea, rezultatul se sumează cu următorul etc.

2. Se citesc de la tastatură n numere întregi. Să se calculeze și afișeze produsul lor, folosind apeluri ale unei funcții ce calculează produsul a câte două numere reale.

3. * Din fișierul laturi.txt se citesc: de pe prima linie numărul natural n, iar de pe fiecare din următoarele n linii câte trei numere reale. Definiți o funcție care verifică dacă trei numere reale pot fi laturile unui triunghi. Folosiți această funcție pentru a afișa numerele de ordine ale triunghiurilor ce se pot forma cu laturile din fișier.
Indicație: funcția va verifica, pentru un triunghi, dacă suma a oricare câte două laturi este mai mare decât a treia latură

4. ** Scrieți un program în limbajul C++ standard care să calculeze și să afișeze cel mai mare divizor comun dintre n numere naturale citite de la tastatură. Se va folosi o funcție care calculează și întoarce în numele ei cel mai mare divizor comun (cmmdc) dintre două numere naturale.
Indicație: se calculează cmmdc dintre primele două numere și se pune în variabila d; se calculează cmmdc dintre d și al treilea număr, care se depune tot în d; se

calculează cmmdc dintre d și al patrulea număr, care se depune tot în d etc.; se calculează cmmdc dintre d și ultimul număr, care se depune tot în d; cmmdc final este valoarea din d.

5. ** Scrieți un program în limbajul C++ standard care să calculeze și să afișeze cel mai mic multiplu comun dintre n numere naturale citite de la tastatură. Se va folosi o funcție care calculează și întoarce în numele ei cel mai mic multiplu comun (cmmmc) dintre două numere naturale.
Indicație: fie două numere naturale a și b, care au un cmmdc și un cmmmc. Între aceste patru numere există relația a*b=cmmdc*cmmmc

6. ** Se citește un număr natural nr de la tastatură. Afișați cel mai apropiat termen Fibonacci de valoarea nr. Se va folosi o funcție care generează termeni ai șirului Fibonacci și calculează distanța termenului generat față de nr, în valoare absolută.

7. ** Descrieți o funcție care verifică dacă un număr dat este palindrom. Folosiți această funcție pentru a afișa toate numerele naturale palindrom mai mici decât numărul natural n dat.
Indicație: Un număr este palindrom dacă citit invers dă aceeași valoare.
Exemple de numere palindrom: 232, 300121003, 76322367

8. * Un tablou unidimensional (vector) se completează cu n numere reale. Să se verifice dacă tabloul este ordonat crescător sau descrescător. Se vor folosi două funcții, fiecare verificând unul din cele două cazuri de mai sus.

9. * Realizați un program în limbajul C++ care compară vechiul algoritm de sortare Bubble-Sort-vechi cu algoritmul Bubble-Sort-optimizat prezentat la Problema 8 din secțiunea Probleme rezolvate.
Indicație: se ia un vector V cu n(n> 5000) elemente generate aleator; se sortează V cu Bubble-Sort-vechi și se observă durata execuției programului; la fel se procedează și cu Bubble-Sort-optimizat aplicat tot pe V original; se compară cele două durate.

10. ** Descrieți două funcții: una care verifică dacă un număr are formă de vârf, cealaltă care verifică dacă un număr are formă de vale. Folosiți cele două funcții pentru a verifica n numere naturale, care pot fi vârf, vale sau oarecare.
Exemple de numere vârf: 234532, 35896321, 2678965, 1372, 232
Exemple de numere vale: 87653479, 97532468, 323, 7689
Exemple de numere oarecare: 34529, 964782, 321, 358, 3456548

11. Se dau numerele naturale a și b. Să se afișeze toate numerele prime din intervalul [a,b]. Se vor folosi apeluri utile ale unei funcții care verifică dacă un număr natural este prim sau nu.

12. *** Fie următorul șir s:
1, 2, 12, 212, 12212, 21212212, 1221221212212, 21212212 1221221212212, …
Se citește de la tastatură un termen t al șirului s de mai sus. Să se afișeze termenul precedent din șirul s. Se va concepe un program cât mai rapid din punct de vedere al duratei de execuție și care utilizează cât mai puțină memorie internă. Programul va folosi exact două funcții: funcția program principal main și încă una impusă de cerințele problemei.
Indicație: Numărul de cifre nrcif al termenului t este un termen al șirului Fibonacci; nrcif se descompune în doi termeni imediat anteriori din Fibonacci; termenul precedent din Fibonacci indică numărul de cifre al termenului precedent din șirul s.
Exemplu: Dacă se dă termenul t=1221221212212, acesta are nrcif=13, care se descompune în termeni Fibonacci astfel 5+8=13. Deci, termenul precedent din șirul s are 8 cifre. Acesta este dat de ultimele 8 cifre din termenul t și este 21212212.
Observație: O altă rezolvare ar putea genera termenii șirului s până când se atinge termenul t. Dar, este ineficientă timp, iar spațiul de memorie utilizat este mare.

13. * Două matrice A și B de dimensiuni n x m se completează cu numere reale citite de la tastatură. Să se calculeze matricea sumă MS și matricea diferență MD. Matricea sumă MS se obține prin adunarea element cu element, iar matricea diferență MD se obține prin diferența element cu element. Apoi să se afișeze matricele MS și MD. Funcția program principal main() va conține numai apeluri de funcții.
Indicație: Se for folosi următoarele funcții fără tip:
- completează – citește numere reale de la tastatură și completează o matrice
- suma – calculează sumele element cu element și construiește matricea MS
- difer - calculează diferențele element cu element și construiește matricea MD
- afis – afișează conținutul unei matrice.

Aceste funcții se apelează în main() astfel:
- completează de două ori, pentru a completa matricele A și B
- suma o dată, pentru a calcula A+B
- diferența o dată, pentru a calcula A-B
- afis de două ori, pentru a afișa matricele MS și MD.

14.** Se citesc n cuvinte formate numai din litere mici ale alfabetului englez. Să se determine dacă sunt anagrame ale aceluiași cuvânt (sunt compuse din aceleași litere, dar în altă ordine).
Indicație: pentru fiecare cuvânt se marchează într-un vector literele folosite și de câte ori apare fiecare literă. Se compară acești vectori doi câte doi. Dacă sunt diferiți, atunci nu sunt anagrame. Dacă sunt la fel, se continuă până la epuizarea tuturor cuvintelor.
Exemplu: fie 4 cuvinte : bechie, cehibe, hibece, cehide

Vectorii de marcaj corespunzători au câte 26 elemente(numărul de litere din tabela ASCII) și sunt următorii:
a b c e f g h i j k…z
011020011000…0
010200110000… 0
011020011000…0
001120011000…0
Observăm că ultimii doi vectori de marcaj sunt diferiți, deci cuvintele nu sunt anagrame.

15.* Scrieți un subprogram care calculează cifra de control a numărului n primit ca parametru de intrare. Se numește cifră de control cifra obținută prin calculul repetat al sumei cifrelor unui număr natural. De exemplu: numărul 75 are cifra de control 3 (7+5=12, 1+2=3) ; 9428 are cifra de control 5 (9+4+2+8=23, 2+3=5). Se citește un vector V cu n elemente numere naturale. Să se afișeze elementele vectorului grupate pe rânduri separate, în ordinea crescătoare a cifrei de control.
Exemplu: dacă numerele sunt 9428 134 75 3401 6051 323, se vor afișa
75 6051 - cifra de control 3
9428 - cifra de control 5
134 323 3401 - cifra de control 8

BIBLIOGRAFIE

1. Clara Ionescu, Gabriela Bălan, Mihaela Giurgea, Claudiu Soroiu – *Informatică pentru grupele de performanță*, Editura Dacia Educațional, Cluj-Napoca, 2004
2. Dan Pracsiu – *Culegere de probleme semnificative de informatică*, Editura Media Sind, Vaslui, martie 2015
3. Dana Lica, Mircea Pașoi – *Informatică. Fundamentele programării*, Editura L&S Soft, 2005
4. Donald E. Knuth – *Arta programării calculatoarelor, vol. 4, fascicola 2, Generarea tuturor tuplurilor și permutărilor*, Editura Teora SRL, București, 2005
5. Emanuela Cherchez, Marinel Șerban – *Programarea în limbajul C/C++ pentru liceu*, Editura Polirom, Iași, 2005
6. Nicolae Cecilian Istrate, Dumitru Fanache, Marius Duță – *Informatica. Manual pentru clasa a X-a*, Editura Gimnasium, Târgoviște, 2000
7. Nicolae Cecilian Istrate, Luminița Duță, Adriana Alexandru, Gabriel Gorghiu – *Programarea calculatoarelor în limbajul C++,* Editura Cetatea de Scaun, Târgoviște, 2008
8. Nicolae Cecilian Istrate și colaboratori, Coordonatori Giorgie Vlad, Ovidiu Marcu – *Informatică. Modele de rezolvare. Bacalaureat 2008*, Editura Gil, Zalău, 2008
9. Răzvan Andonie, Ilie Gârbacea – *Algoritmi fundamentali. O perspectivă C++*, Editura Libris, Cluj, 1995
10. Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest – *Introducere în algoritmi*, Editura Computer Libris Agora, Cluj Napoca, 2000
11. Tudor Sorin – *Tehnici de programare, manual pentru clasa a X-a*, Editura L&S Infomat, 1996
12. Site Lista lui Frâncu, http://probleme.francu.com/rom/home.html , accesat la 22 iunie 2015
13. Site Campion-pregătire olimpici, http://campion.edu.ro/arhiva, accesat la 22 iunie 2015
14. Site Infoarena-pregătire olimpici, http://infoarena.ro, accesat la 22 iunie 2015
15. Site Gazeta de informatică, http://gazeta.info.ro/ , accesat la 22 iunie 2015