Laborator 4
Alocarea dinamică a memoriei

În acest capitol sunt prezentate consideraţii teoretice privind alocarea dinamică a memoriei şi utilizarea funcţiilor predefinite în C : malloc(), calloc(), realloc, free(), etc. şi a noilor funcţii new şi delete introduse în C++.

CONSIDERAŢII TEORETICE

Cele mai importante **funcţii** pentru **alocarea dinamică a memoriei sunt :**

a) în C : malloc(), realloc(), calloc(), free(), etc.

b) în C++: toate cele din C și adițional se utilizează operatorii new şi delete.

Funcţia malloc() are următorul prototip:

void *malloc(unsigned n);

unde:

n = numărul de octeţi de memorie ce va fi alocat dinamic.

La apelare, funcţia malloc() returnează un pointer spre primul octet al regiunii de memorie alocate în memoria (heap) liberă. Funcţia alocă în memoria heap o zonă contiguă de n octeţi = numărul de octeţi specificat în paranteza funcţiei malloc().

Dacă nu este suficientă memorie disponibilă atunci malloc() returnează “null”.
Bibliotecile ce trebuie incluse în program pentru utilizarea acestei funcţii sunt: <stdlib.h> şi <malloc.h>

Alocarea unei zone de memorie contigue, neinițializate se realizează diferit în C sau C++ după cum este ilustrat în cele ce urmează.

a) secvenţa în C

```
tip *p;  p=malloc(n*sizeof(tip);
```

b) secvenţa în C++

```
tip *p; p=(tip*)malloc(n*sizeof(tip);
```

În C nu este necesară specificarea tipului pentru a atribui lui p valoarea returnată de malloc(), deoarece un pointer de tip *void este automat convertit în tipul pointerului din partea stângă a atribuirii. În C++ este obligatorie specificarea explicită de tip atunci când se atribuie un pointer de tip *void altui tip de pointer. O altă observație importantă legată de utilizarea funcţiei malloc() este că zona de memorie alocată nu este inițializată.

Pentru evitarea erorilor trebuie inclus în program un test din care să rezulte dacă există memorie disponibilă sau nu (“null”). Testarea valorii returnate de funcţia malloc(), deci a existenţei unei zone de memorie libere, se poate realiza ca şi în exemplu următor:

Ex.: alocarea dinamica a unei zone de memorie pentru 100 de numere întregi =100*4 octeți=400

octeți. Se testează dacă există memorie liberă, iar dacă nu există se afişează mesajul de eroare „memorie insuficienta”:

```
int *ip = malloc(100 * sizeof(int));
if(ip == NULL)
   {printf("memorie insuficienta\n");
    exit(1)     }
```

O altă funcţie importantă este **funcţia free()** reprezentând funcţia opusă funcţiei malloc() care are ca efect eliberarea memoriei alocate dinamic anterior cu funcţia malloc().

Funcţia free() are următorul prototip: **void *free(void *p);**

unde: **p** este un pointer spre memoria alocată anterior cu funcţia malloc().

Bibliotecile ce trebuie incluse în program pentru utilizarea funcţiei free() sunt : <stdlib.h> si <malloc.h>. Un exemplu de utilizare a funcţiei free() este prezentat mai jos.

Ex.: Alocarea și dezalocarea unei zone de memorie pentru 50 de numere întregi:

```
int *p;
p= (int *)malloc(50*sizeof(int));
…
free(*p);
```

Functia realloc() are următorul prototip: **void *realloc(void *p, unsigned n);**

unde: **p** este un pointer spre memoria realocată dinamic

Funcţia realloc() realocă dinamic zona de memorie specificată prin numărul de octeți, n, spre care indică pointerul p .
Bibliotecile ce trebuie incluse în program pentru utilizarea acestei funcţii sunt: <stdlib.h> şi <malloc.h>

Ex: Iniţial se alocă dinamic memorie pentru 50 de numere întregi . Ulterior se realocă dinamic o zona de memorie pentru 100 de numere întregi , acelaşi pointer p conţinând adresa de început a zonei alocate dinamic:

```
int *p;
p= (int *)malloc(50*sizeof(int));
…
p= (int *)realloc(p, 100 * sizeof(int));
```

Dacă realocarea nu este posibilă din lipsa de memorie libera atunci realloc() returnează ‘NULL’, altfel funcția returnează un pointer spre zona de memorie realocată.

Funcţia calloc() este **funcția de alocare și inițializare cu 0 a memoriei** şi are următorul prototip:

void *calloc(unsigned nrelem, unsigned dimelem);

unde

nrelem=numărul de obiecte pentru care se rezervă memorie

dimelem= numărul de octeţi de memorie ce va fi alocat pentru fiecare element

Efectul apelării acestei funcţii este alocarea unei zone de memorie de dimensiune nrelem*dimelem, în memoria heap care se inițializează cu 0, funcţia returnând un pointer la zona de memorie sau 'null' dacă alocarea nu s-a putut realiza.

Bibliotecile ce trebuie incluse în program pentru utilizarea funcţiei calloc sunt: <stdlib.h> şi <malloc.h>

Un exemplu de utilizare a acestei funcţii este prezentat în continuare:

Ex.: alocarea unei zone de memorie pentru 100 de numere întregi iniţializate cu 0 :

```
int *p;
p = (int *) calloc(100, sizeof(int));
```

Eliberarea zonei alocate dinamic cu funcţia calloc() se poate face de asemenea utilizând funcţia free() .

Alocarea dinamică a memoriei în C++ se realizează cel mai frecvent prin intermediul operatorilor **new** şi **delete.**

Formatul de declarare al operatorului **new** este :

new tip sau **new (tip)** sau
new tip (expresie) sau **new tip[exp]**

unde: **tip** = numele unui tip predefinit sau definit de utilizator

expresie = expresie a cărei valoare inițializează zona de memorie alocată prin new

exp = expresie de tip int folosită la alocarea dinamică a tablourilor.

Operatorul **new** permite alocarea memoriei în zona heap şi are ca şi valoare adresa de început a zonei de memorie alocate sau 0 dacă alocarea eşuează . Operatorul **delete** este utilizat pentru eliberarea zonei de memorie alocată cu operatorul new. Un exemplu de utilizare al operatorilor new și delete este prezentat mai jos.

Ex.:

```
tip *p;
p=new tip;
…
delete p;
```

Un alt exemplu de utilizare al acestor operatori, cu alocarea/dealocarea memoriei pentru un tablou este prezentat mai jos:

Ex.:

```
tip *p=new tip[exp];          //alocarea unui tablou
delete [exp] p;               //eliberarea memoriei ocupate de tablou
```

PROBLEME REZOLVATE

Ex.1: Programul realizează alocarea dinamică a memoriei pentru un șir de n numere întregi, unde n este un număr întreg citit de la tastatură și calculează suma și produsul elementelor șirului, precum și elementul minim și maxim din sir.

Varianta in C		Varianta in C++
#include <stdio.h> #include <malloc.h> //pentru functia malloc() int main() {int *p, n,i,sum=0, prod=1, max, min; printf("Introduceti n intreg pozitiv: ");scanf("%d", &n); **//alocarea dinamica** p=(int*)malloc(n*sizeof(int)); if(p==NULL) printf("\n eroare de alocare!"); printf("Introduceti %d nr. intregi:\n", n); for (i=0;i<n;i++) {printf("nr %d:",i+1); scanf("%d", p+i); } **//afisarea sirului de nr** printf("ati introdus nr:"); max=min=*p; for (i=0;i<n;i++) {printf("%d ",*(p+i)); sum+=*(p+i);prod*=*(p+i); if (max<*(p+i)) max=*(p+i); if (min>*(p+i)) min=*(p+i);} printf("\nSuma elementelor =%d\n", sum); printf("Produsul elementelor=%d\n",prod); printf("maxim=%d\n", max); printf("minim=%d\n", min); **//eliberarea memoriei** free (p); return 0;}		#include <iostream> #include <malloc.h> using namespace std; int main() {int *p, n,i,sum=0, prod=1, max, min; cout<<"Introduceti n intreg pozitiv: ";cin>>n; **//alocarea dinamica** p=new int[n]; if(p==NULL) cout<<endl<<" eroare de alocare!"; cout<<"Introduceti"<<n<<"nr. intregi:"<<endl; for (i=0;i<n;i++) {cout<<"nr"<<i+1<<":"; cin>>*(p+i); } **//afisarea sirului de nr** cout<<"ati introdus nr:"; max=min=*p; for (i=0;i<n;i++) {cout<<*(p+i)<<" "; sum+=*(p+i);prod*=*(p+i); if (max<*(p+i)) max=*(p+i); if (min>*(p+i)) min=*(p+i);} cout<<"\nSuma elementelor ="<<sum<<endl; cout<<"Produsul elementelor="<<prod<<endl; cout<<"maxim="<<max<<endl; cout<<"minim="<<min<<endl; **//eliberarea memoriei** delete p; return 0;}

Rezultate:

```
Introduceti n intreg pozitiv: 5
Introduceti 5 nr. intregi:
nr 1:5
nr 2:10
nr 3:9
nr 4:4
nr 5:3
ati introdus nr:5 10 9 4 3
Suma elementelor =31
Produsul elementelor=5400
maxim=10
minim=3
```

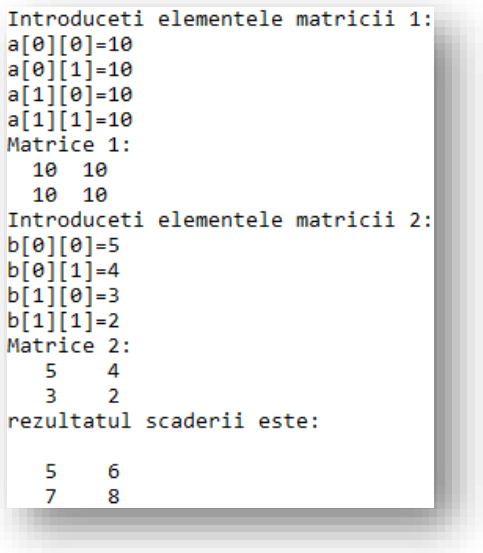
Aplicație:

Să se rescrie programul in C++ astfel încât să se utilizeze operatorii malloc și free pentru alocarea dinamică a memoriei și efectuarea calculelor și afișării rezultatelor.

Ex2.: Programul realizează scăderea a două matrici pătratice de aceeași dimensiune n, unde n este un număr întreg introdus de la tastatură, utilizând alocarea dinamică a memoriei pentru cele două matrici .

Varianta in C		Varianta in C++
#include <stdio.h> #include <stdlib.h> #include<malloc.h> int main() {int n,i,j, *p,*q; printf("Introduceti dimensiunea matricilor patratice, n="); scanf("\n%d",&n); if((p=(int*)malloc(n*n*sizeof(int)))==NULL) { printf("eroare alocare memorie matrice 1\n"); } if((q=(int*)malloc(n*n*sizeof(int)))==NULL) { printf("eroare alocare memorie matrice 2\n"); } **//Citirea elementelor primei matrici** printf("\nIntroduceti elementele matricii 1:\n"); for(i=0;i<n;i++) for(j=0;j<n;j++) { printf("a[%d][%d]=",i,j); scanf("%d",p+i*n+j); } **//tiparirea elementelor primei matrici** printf("Matrice 1:"); for(i=0;i<n;i++) { printf("\n"); for(j=0;j<n;j++) printf("%4d",*(p+i*n+j));} **//citirea elementelor matricii a doua** printf("\nIntroduceti elementele matricii 2:\n"); for(i=0;i<n;i++) for(j=0;j<n;j++) { printf("b[%d][%d]=",i,j); scanf("%d",q+i*n+j);} **//tiparirea elementelor matricii a doua** printf("Matrice 2:"); for(i=0;i<n;i++){ printf("\n"); for(j=0;j<n;j++) printf("%4d ",*(q+i*n+j)); } printf("\n"); **//scaderea matricilor si tiparirea rezultatului** printf("rezultatul scaderii este:\n"); for(i=0;i<n;i++){ printf("\n"); for(j=0;j<n;j++) printf("%4d ",*(p+i*n+j)- *(q+i*n+j));} free(p);free(q); return 0;}		#include <iostream> #include <stdlib.h> #include<malloc.h> using namespace std; int main() {int n,i,j, *p,*q; cout<<"Introduceti dimensiunea matricilor patratice, n="; cin>>n; p=new int[n*n]; if(p==NULL) cout<<"eroare alocare memorie matrice 1"<<endl; q=new int[n*n]; if(q==NULL) cout<<"eroare alocare memorie matrice 2"<<endl; **//Citirea elementelor primei matrici** cout<<"\nIntroduceti elementele matricii 1:"<<endl; for(i=0;i<n;i++) for(j=0;j<n;j++) { cout<<"a["<<i<<"]["<<j<<"]="; cin>>*(p+i*n+j); } **//tiparirea elementelor primei matrici** cout<<"Matrice 1:"; for(i=0;i<n;i++) { cout<<endl; for(j=0;j<n;j++) cout<<*(p+i*n+j)<<"\t";} **//citirea elementelor matricii a doua** cout<<"\nIntroduceti elementele matricii 2:"<<endl; for(i=0;i<n;i++) for(j=0;j<n;j++) { cout<<"b["<<i<<"]["<<j<<"]="; cin>>*(q+i*n+j);} **//tiparirea elementelor matricii a doua** cout<<"Matrice 2:"; for(i=0;i<n;i++){ cout<<endl; for(j=0;j<n;j++) cout<<*(q+i*n+j)<<"\t"; } cout<<endl; **//scaderea matricilor si tiparirea rezultatului** cout<<"rezultatul scaderii este:"<<endl; for(i=0;i<n;i++){ cout<<endl; for(j=0;j<n;j++) cout<<*(p+i*n+j)- *(q+i*n+j)<<"\t";} delete p;delete q; return 0;}

Rezultate:



```
Introduceti elementele matricii 1:
a[0][0]=10
a[0][1]=10
a[1][0]=10
a[1][1]=10
Matrice 1:
  10  10
  10  10
Introduceti elementele matricii 2:
b[0][0]=5
b[0][1]=4
b[1][0]=3
b[1][1]=2
Matrice 2:
   5    4
   3    2
rezultatul scaderii este:

   5    6
   7    8
```

Aplicație:

Să se modifice programul astfel încât să se calculeze și afișeze matricea rezultată din expresia:
10*matrice1-2*matrice2.

Ex.3: Programul realizează citirea și afișarea unei matrici cu elemente întregi, de dimensiune nxn, unde n este număr întreg introduse de la tastatură, și realizează înmulțirea sa cu o constantă întreagă citită de la tastatură. După afișarea rezultatului se cere să se realoce dinamic memorie pentru un șir de n numere întregi ce vor fi citite de la tastatură și apoi afișate la puterea a 3-a.

Varianta in C		**Varianta in C++**
#include <stdio.h> #include <malloc.h> #include <math.h> int main() {int n,i,j,*p,x,y; printf("Introduceti dimensiunea matricilor patratice, n="); scanf("\n%d",&n); **//alocarea dinamica si testare pointer** if(p=(int*)malloc(n*n*sizeof(int))) {printf("Introduceti elementele matricii:\n"); for (i=0;i<n;i++) for (j=0;j<n;j++) {printf("a[%d][%d]=",i,j); scanf("%d", p+i*n+j);} **//afisarea matricii** printf("matricea este:\n"); for (i=0;i<n;i++) {for (j=0;j<n;j++) printf("%4d",*(p+i*n+j)); printf("\n");} printf("introduceti un nr intreg x="); scanf("%d",&x); printf("matricea inmultita cu %d este:\n",x); for (i=0;i<n;i++) {for (j=0;j<n;j++) printf("%4d",(*(p+i*n+j)*x)); printf("\n"); }} else {printf("eroare de alocare memorie!");} printf("\nIntroduceti un sir, de dimensiune n= "); scanf("%d", &n); p=(int*)realloc(p,n*sizeof(int)); if(p==NULL) printf("\n eroare de alocare!");		#include <iostream> #include <malloc.h> #include <math.h> using namespace std; int main() {int n,i,j,*p,x,y; cout<<"Introduceti dimensiunea matricilor patratice, n="; cin>>n; **//alocarea dinamica si testare pointer** p=(int*)malloc(n*n*sizeof(int)); if(p!=NULL) {cout<<"Introduceti elementele matricii:"<<endl; for (i=0;i<n;i++) for (j=0;j<n;j++) {cout<<"a["<<i<<"]["<<j<<"]="; cin>>*(p+i*n+j);} **//afisarea matricii** cout<<"matricea este:"<<endl; for (i=0;i<n;i++) {for (j=0;j<n;j++) cout<<*(p+i*n+j)<<"\t"; cout<<endl;} cout<<"introduceti un nr intreg x="; cin>>x; cout<<"matricea innultita cu "<<x<<" este:"<<endl; for (i=0;i<n;i++) {for (j=0;j<n;j++) cout<<*(p+i*n+j)*x<<"\t"; cout<<endl;}}

```
printf("Introduceti elementele sirului:\n");
for (i=0;i<n;i++)
{printf("a[%d]=",i); scanf("%d", p+i);
}
printf("sirul la puterea 3-a:\n");
for (i=0;i<n;i++)
{y=pow(*(p+i),3);
printf("\na[%d]^3=%d",i,y);}
printf("\n");
free (p); //eliberarea memoriei
return 0;}
```

```
else {cout<<"eroare de alocare memorie!";}
cout<<"\nIntroduceti un sir, de dimensiune n= ";
cin>>n;
p=(int*)realloc(p,n*sizeof(int));
if(p==NULL) cout<<"\n eroare de alocare!";
cout<<"Introduceti elementele sirului:"<<endl;
for (i=0;i<n;i++)
{cout<<"a["<<i<<"]="; cin>>*(p+i);
}
cout<<"sirul la puterea 3-a:"<<endl;
for (i=0;i<n;i++)
{y=pow(*(p+i),3);
cout<<endl<<"a["<<i<<"]^3="<<y;}
cout<<endl;
free (p); //eliberarea memoriei
return 0;}
```

Rezultate:

```
a[0][0]=1
a[0][1]=2
a[1][0]=3
a[1][1]=4
matricea este:
1       2
3       4
introduceti un nr intreg x=2
matricea innultita cu 2 este:
2       4
6       8

Introduceti un sir, de dimensiune n= 5
Introduceti elementele sirului:
a[0]=10
a[1]=10
a[2]=2
a[3]=5
a[4]=1
sirul la puterea 3-a:

a[0]^3=1000
a[1]^3=1000
a[2]^3=8
a[3]^3=124
a[4]^3=1
```

Aplicație:

Să se modifice programul astfel încât să se calculeze și afișeze maximul și minimul din șirul realocat dinamic.

Ex.4. Programul realizează înmulțirea a două matrici de numere întregi, prima matrice de dimensiune mxn (m linii și n coloane) iar cea de-a doua matrice de nxl (n linii și l coloane) utilizând alocarea dinamică a memoriei.

Varianta in C		Varianta in C++
#include <stdio.h> #include <malloc.h> int main() {int n,m,i,s,j,k,l,*p,*q, a[10][10],b[10][10]; p=&a[0][0]; q=&b[0][0]; printf("nr. de linii prima matrice,m=");scanf("%d",&m); printf("\nnr. de coloane prima matrice=\nnr. de linii a doua matrice,n=");		#include <iostream> #include <malloc.h> using namespace std; int main() {int n,m,i,s,j,k,l,*p,*q,a[10][10],b[10][10]; p=&a[0][0]; q=&b[0][0]; cout<<"nr. de linii prima matrice,m=";cin>>m; cout<<"\nnr. de coloane prima matrice=\nnr. de linii a doua matrice,n="; cin>>n;

```
scanf("%d",&n);
printf("nr de coloane la a doua matrice
l=");scanf("%d",&l);
if((p=(int*)malloc(m*n*sizeof(int)))==NULL)
{printf("nu exista suficienta memorie\n");}
printf("\n Prima matrice:\n");
for (i=0;i<m;i++)
  for (j=0;j<n;j++)
  {  printf("a[%d][%d]=",i+1,j+1); scanf("%d",*(a+i)+j);   }
if((q=(int*)malloc(n*l*sizeof(int)))==NULL)
{printf("nu exista suficienta memorie\n");}
printf("\n A doua matrice:\n");
for (i=0;i<n;i++)
  for (j=0;j<l;j++)
  {  printf("b[%d][%d]=",i+1,j+1);scanf("%d",*(b+i)+j);  }
    printf("\nMatricea produs este");
  for (i=0;i<m;i++)
  {  printf("\n");
     for (j=0;j<l;j++)
     { s=0;
       for (k=0;k<n;k++)
       s+=*(*(a+i)+k)* (*(*(b+k)+j));  printf("%4d  ",s);}}
printf("\n");
free(p);free(q);
return 0;}
```

```
cout<<"nr de coloane la a doua matrice l=";cin>>l;
p=new int[m*n];
if(p==NULL)
{cout<<"nu exista suficienta memorie"<<endl;}
cout<<endl<<"Prima matrice:"<<endl;
for (i=0;i<m;i++)
  for (j=0;j<n;j++)
  {  cout<<"a["<<i+1<<"]["<<j+1<<"]=";
cin>>*(*(a+i)+j);   }
q=new int[n*l];
if(q==NULL)
{cout<<"nu exista suficienta memorie"<<endl;}
cout<<endl<<" A doua matrice:"<<endl;
for (i=0;i<n;i++)
  for (j=0;j<l;j++)
  {  cout<<"b["<<i+1<<"]["<<j+1<<"]=";
cin>>*(*(b+i)+j); }
    cout<<endl<<"Matricea produs este";
  for (i=0;i<m;i++)
  {  cout<<endl;
     for (j=0;j<l;j++)
     { s=0;
       for (k=0;k<n;k++)
       s+=*(*(a+i)+k)* (*(*(b+k)+j));
cout<<s<<"\t";}}
cout<<endl;
delete p; delete q;
return 0;}
```

Rezultate:

```
nr. de linii prima matrice,m=2

nr. de coloane prima matrice=
nr. de linii a doua matrice,n=3
nr de coloane la a doua matrice l=2

 Prima matrice:
a[1][1]=1
a[1][2]=2
a[1][3]=3
a[2][1]=1
a[2][2]=2
a[2][3]=3

 A doua matrice:
b[1][1]=2
b[1][2]=2
b[2][1]=2
b[2][2]=10
b[3][1]=10
b[3][2]=10

Matricea produs este
  36    52
  36    52
```

Aplicație:

Să se modifice programul astfel încât să se calculeze și afișeze matricile la pătrat

Ex.5. Programul (în C++) alocă memorie dinamică pentru trei șiruri de câte 256 de caractere și eliberează memoria alocată dinamic după inițializarea celor trei șiruri cu valori constante.

Varianta 1 in C++		Varianta 2 in C++
#include <stdio.h> #include <stdlib.h> int main() {char *sir1=new char[10]; char *sir2,*sir3; int i; sir2=new char[10]; for (i=0;i<10;i++) {sir1[i]='A'; printf("%c%d ",sir1[i], i); sir2[i]='Z'; printf("%c%d ",sir2[i],i);} delete sir1; printf("\n"); sir3=new char[10]; for (i=0;i<10;i++) {sir3[i]=sir2[i]; printf("%c ",sir3[i]);} printf("\n"); delete sir2;delete sir3; return 0;}		#include <iostream> #include <stdlib.h> using namespace std; int main() {char *sir1=new char[10]; char *sir2,*sir3; int i; sir2=new char[10]; for (i=0;i<10;i++) {sir1[i]='A'; cout<<sir1[i]<<i<<"\t"; sir2[i]='Z'; cout<<sir2[i]<<i<<"\t";} delete sir1; cout<<endl; sir3=new char[10]; for (i=0;i<10;i++) {sir3[i]=sir2[i]; cout<<sir3[i]<<" ";} cout<<endl; delete sir2;delete sir3; return 0;}

Rezultate:

```
A0 Z0 A1 Z1 A2 Z2 A3 Z3 A4 Z4 A5 Z5 A6 Z6 A7 Z7 A8 Z8 A9 Z9
Z Z Z Z Z Z Z Z Z Z
```

Aplicație:

Să se modifice programul astfel încât să se calculeze și afișeze elementele șirurilor din 2 în 2.

Ex.6. Programul (în C++) alocă memorie dinamică pentru o matrice de nxm elemente intregi citite de la tastatura utilizand new si afiseaza matricea si matricea inmultita cu o constanta intreaga.

Varianta 1 in C++		Varianta 2 in C++
#include <stdio.h> int main() {int n, m, ** mat, i, j, c; printf("\nNumarul de linii:"); scanf("%d", &n); printf("\nNumarul de coloane:"); scanf("%d", &m); mat=new int*[n]; for(i=0 ; i<n ; i++) mat[i]=new int[m]; //citirea elementelor matricei for(i=0 ; i<n ; i++) for(j=0 ; j<m ; j++) { printf("mat[%d][%d] = ", i, j); scanf("%d", &mat[i][j]); } //afişarea elementelor matricei for(i=0 ; i<n ; i++) { printf("\n"); for(j=0 ; j<m ; j++) printf("%5d", mat[i][j]); } printf("\nConstanta:"); scanf("%d", &c);		#include <iostream> #include <iomanip> using namespace std; int main() {int n, m, ** mat, i, j, c; cout <<"\nNumarul de linii:"; cin>>n; cout <<"\nNumarul de coloane:"; cin >>m; mat=new int*[n]; for(i=0 ; i<n ; i++) mat[i]=new int[m]; //citirea elementelor matricei for(i=0 ; i<n ; i++) for(j=0 ; j<m ; j++) { cout <<"mat"<<"["<<i<<"]["<<j<<"]"; cin>>mat[i][j];} //afişarea elementelor matricei for(i=0 ; i<n ; i++) { cout<<"\n"; for(j=0 ; j<m ; j++) cout<<setw(5)<<mat[i][j]; } cout<<"\nConstanta:"; cin>>c;

```
printf("\nMatricea inmultita cu o constanta:\n");
 for(i=0 ; i<n ; i++)
 { printf("\n"); for(j=0 ; j<m ; j++)
printf("%5d", c*mat[i][j]); }
for(i=0; i<n ; i++) // dealocarea memoriei
 delete [ ] mat[i];
 delete [ ] mat; return 0;}
```

```
 cout<<"\nMatricea inmultita cu o constanta:\n";
 for(i=0 ; i<n ; i++)
 { cout<<"\n"; for(j=0 ; j<m ; j++)
cout<<setw(5)<<c*mat[i][j]; }
for(i=0; i<n ; i++) // dealocarea memoriei
delete [ ] mat[i];
delete [ ] mat;
return 0;}
```

Rezultate:

```
Numarul de linii:2

Numarul de coloane:3
mat[0][0]1
mat[0][1]2
mat[0][2]3
mat[1][0]4
mat[1][1]5
mat[1][2]6

    1    2    3
    4    5    6
Constanta:10

Matricea inmultita cu o constanta:

   10   20   30
   40   50   60
```

Aplicație:

Să se modifice programul astfel încât să se calculeze și afișeze diagonala principala a matricii.

Ex.7. Programul realizeaza cautarea unei litere intr-un bloc de memorie in care s-au stocat valorile unui sir de caractere

Varianta in C

```
#include <stdio.h>
#include <string.h>
int main ()
{char *pch;  char str[] = "Exemplu sir pointeri";
pch = (char*) memchr (str, 'p', strlen(str));
printf("sirul este: %s\n", str);
if (pch!=NULL)
  printf ("'p' apare prima data pe pozitia %d din sir.\n",
pch-str+1);
   else printf (" 'p' nu a fost gasit in sir.\n");
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{char *pch;  char str[] = "Exemplu sir pointeri";
pch = (char*) memchr (str, 'p', strlen(str));
cout <<"sirul este:"<<str;
if (pch!=NULL)
   cout<<"\n'p' apare prima data pe pozitia "<< pch-
str+1<<" din sir.\n";
   else cout <<" 'p' nu a fost gasit in sir.\n";
   return 0;}
```

Rezultate:

```
sirul este:Exemplu sir pointeri
'p' apare prima data pe pozitia 5 din sir.
```

Aplicație:

Să se modifice programul astfel încât să se citeasca sirul de la tastatura si sa se afișeze pe ce pozitie apare litera i prima data in sir.

Ex.8. Programul compara doua blocuri de memorie care contin fiecare cate un sir de caractere si afieaza un mesaj daca sunt identice.

Varianta in C		Varianta in C++
#include <stdio.h>		#include <iostream>

```c
#include <string.h>
int main ()
{char buffer1[] = "Sir text caractere";
 char buffer2[] = "Sir text caractere";
 int n;  n=memcmp( buffer1, buffer2, sizeof(buffer1) );
 if (n>0) printf ("'%s' este mai mare decat '%s'.\n",buffer1,buffer2);
   else if (n<0)
        printf ("'%s' este mai mic decat'%s'.\n",buffer1,buffer2);
        else printf ("'%s' este identic cu '%s'.\n",buffer1,buffer2);
return 0;}
```

```c
#include <string.h>
using namespace std;
int main ()
{char buffer1[] = "Sir text caractere";
 char buffer2[] = "Sir text caractere";
 int n;  n=memcmp( buffer1, buffer2, sizeof(buffer1) );
 if (n>0)
   cout<<buffer1<<" este mai mare decat "<<buffer2;
  else if (n<0)
        cout <<buffer1 <<" este mai mic decat"<<buffer2<<endl;
        else cout<<buffer1<< " este identic cu "<<buffer2<<endl;
return 0;}
```

Rezultate:

```
'Sir text caractere' este identic cu 'Sir text caractere'.
```

Aplicație:

Să se modifice programul astfel încât să se caute in primul sir litera p si litera i si daca se gasesc sa se afiseze pozitia respectiva in sir, utilizand memchr()

Ex.9. Copierea unui bloc de memorie peste alt bloc- copiere sir peste alt sir

Varianta in C

```c
#include <stdio.h>
#include <string.h>
int main ()
{char str1[] = "Test"; char str2[] = "PCLP2";
puts("str1 inainte de copiere cu memcpy ");
puts(str1);/* Copies contents of str2 to str1 */
memcpy (str1, str2, sizeof(str2));
puts("\nstr1 dupa copiere cu memcpy ");
puts(str1);
return 0;}
```

Varianta in C++

```c
#include <iostream>
#include <string.h>
using namespace std;
int main ()
{char str1[] = "Test"; char str2[] = "PCLP2";
cout<<"str1 inainte de copiere cu memcpy "<<endl;
cout<<str1;/* Copies contents of str2 to str1 */
memcpy (str1, str2, sizeof(str2));
cout<<"\nstr1 dupa copiere cu memcpy "<<endl;
cout<<str1;
return 0;}
```

Rezultate:

```
str1 inainte de copiere cu memcpy
Test

str1 dupa copiere cu memcpy
PCLP2
```

Aplicație:

Să se modifice programul astfel încât să se afiseze si str2 dupa copiere.

Ex.10. Mutarea unui bloc de memorie la adresa altui bloc- copiere sir peste alt sir

Varianta in C		Varianta in C++
#include <stdio.h> #include <string.h> int main () {char str1[] = "Test"; char str2[] = "Programare PCLP2"; puts("str1 inainte de memmove "); puts(str1); **memmove(str1, str2, sizeof(str2));** puts("\nstr1 dupa memmove "); puts(str1); return 0;}		#include <iostream> #include <string.h> using namespace std; int main () {char str1[] = "Test"; char str2[] = "Programare PCLP2"; cout<<"str1 inainte de memmove "<<endl; cout<<str1; **memmove(str1, str2, sizeof(str2));** cout<<"\nstr1 dupa memmove "<<endl; cout<<str1; return 0;}

Rezultate:

```
str1 inainte de memmove
Test

str1 dupa memmove
Programare PCLP2
```

Aplicație:

Să se modifice programul astfel încât să se afiseze str2 dupa copiere.

Ex.11. Initializarea unui bloc de memorie care contine un sir de caractere, prin inlocuirea a 10 caractere cu '.' incapand cu pozitia a 10 –a din sir

Varianta in C++		Varianta in C++
#include <stdio.h> #include <string.h> int main() {char str[50] = "Exemplu de utilizare memset"; printf("Inainte de memset(): %s\n", str); // initializeaza 10 octeti cu '.' incepand cu str[10] **memset(str + 10, '.', 10*sizeof(char));** printf("Dupa memset(): %s", str); return 0; }		#include <iostream> #include <string.h> using namespace std; int main() {char str[50] = "Exemplu de utilizare memset"; cout<<"Inainte de memset(): "<< str<<endl; // initializeaza 10 octeti cu '.' incepand cu str[10] memset(str + 10, '.', 10*sizeof(char)); cout<<"Dupa memset(): "<< str; return 0;}

Rezultate:

```
Inainte de memset(): Exemplu de utilizare memset
Dupa memset(): Exemplu de.......... memset
```

Aplicație:

Să se modifice programul astfel încât să inlocuiasca caracterul . cu *

PROBLEME PROPUSE

1. ***Să se scrie programul care realizează alocarea dinamică a memoriei pentru un şir de n (citit de la tastatura) numere reale şi calculează şi afişează termenii şirului şi transpusa unei matrici utilizând alocarea dinamică a memoriei.***

2. ***Să se scrie programul care realizează inversa şi transpusa unei matrici utilizând alocarea dinamică a memoriei.***

3. ***Se consideră o matrice A cu m linii şi n coloane, cu elemente reale. Să se scrie programul care calculează şi afişează expresia de mai jos utilizând alocarea dinamică a memoriei:***

$$E = \sqrt{\prod_{i=1}^{m} \left(\sum_{j=1}^{n} a_{i,j}^2 \right)}$$

, unde $a_{i,j}$ ***sunt elementele matricii.***

1. Sa se scrie un program care citeste de la tastatura un sir x[i] , i=1, n (n intreg) numere reale şi calculeaza si afiseaza valoarea functiei : $f(x) = \begin{cases} x^2 - 1, x > 0 \\ -5, x < 0 \end{cases}$***, utilizand alocare dinamica a memoriei.***