Anexa 1

FUNCŢII DE PRELUCRARE A STRUCTURILOR DE DATE

Masive

Problema MS1: să se însumeze elementele a trei vectori, fiecare a câte *n* elemente, definite într-o zonă de memorie contiguă.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int vect[10];
const vect x={ 1,2,3,4,5,6,7,8,9,10 };
const vect y={ 10,9,8,7,6,5,4,3,2,1 };
const vect z={ 1,1,2,2,3,3,4,4,5,5 };
int s=0;
void main(){
   clrscr();
   for(int i=0;i<30;i++) s+=x[i];
   printf("\nSuma este: %d ",s);
   getch();
   }
```

Problema MS2: se consideră un vector de maxim 20 de elemente de tip întreg. Numărul de elemente şi valorile lor se citesc de la terminal. Să se ordoneze crescător elementele vectorului şi să se scrie funcţia de inserare a unui element în vector, astfel încât, după operaţia de inserare vectorul să rămână ordonat.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int vector[20];
vector a;
int n;
void sortare(vector x,int n){
  int temp;
  for(int i=0;i<n-1;i++)
     for(int j=i+1;j<n;j++)
         if (x[i]>x[j]){
                    temp=x[i];
                    x[i]=x[j];
                    x[j]=temp;
                       }
}
void tiparire(vector x,int n)
{ for(int i=0;i<n;i++)
  printf("%5d",x[i]);
}
int inserare(vector x,int n)
 { int poz;
   int elem;
   if (n==19)
     printf("\nVector plin!");
```

```
     else
       { printf("\nDati elem. de inserat: ");
      scanf("%d",&elem);
     if (elem<=x[0])
          poz=0;
     else
          if (elem>=x[n-1])
               poz=n;
          else
               for(int i=0;i<n-1;i++)
                  if ((x[i]<=elem)&&(elem<x[i+1]))
                        poz=i+1;
         }
     for(int i=n;i>=poz;i--)
              x[i]=x[i-1];
     x[poz]=elem;
     n++;
     return n;
  }
void main()
{ clrscr();
  printf("\nDati dimensiunea vectorului : n="); scanf("%d",&n);
  printf("\nIntroduceti elementele vectorului: \n");
  for(int i=0;i<n;i++)
      { printf("a(%d)=",i);
       scanf("%d",&a[i]);
      };
  printf("\nVectorul initial este: ");
  tiparire(a,n);
  sortare(a,n);
  printf("\nVectorul dupa ordonare este: ");
  tiparire(a,n);
  n=inserare(a,n);
  printf("\nVectorul dupa inserare este: ");
  tiparire(a,n);
  printf("\nNumarul de elem. a vectorului: %d",n);
  getch();
}
```

Problema MS3: se consideră un vector de elemente de tip întreg de la problema precedentă. Să se sorteze crescător elementele acestui vector şi să se scrie două variante de ştergere a unui element din vector :

- indicând poziţia acestuia în vector;
- după o valoare citită de la terminal;

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef int vect[20];
vect x;
int elem,n,poz;
void sortare(int x[20],int n)
{
  int temp;
  for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
       if (x[i]>x[j])
        { temp=x[i];
          x[i]=x[j];
```

```
          x[j]=temp;
        }
}
void tiparire(int x[20],int n)
{ for(int i=0;i<n;i++)
  printf("%5d",x[i]);
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. a vectorului: "); scanf("%d",&n);
  printf("\nIntroduceti elem. vectorului:\n ");
  for(int i=0;i<n;i++)
  scanf("%d",&x[i]);
  sortare(x,n);
  printf("\nVectorul sortat este: ");
  tiparire(x,n);
  printf("\nDati pozitia elem. de sters: "); scanf("%d",&poz);
  for(i=poz-1;i<n-1;i++)
     x[i]=x[i+1];
  n--;
  printf("\nVectorul devine: ");
  tiparire(x,n);
//var.2
  printf("\nDati elem. pe care doriti sa-l stergeti: ");
scanf("%d",&elem);
//verificam daca val. citita este sau nu in vector
  poz=-1;
  for(i=0;i<n;i++)
  if(x[i]==elem)
        poz=i;
  if (poz==-1)
        printf("\nNu exista in vector aceasta val.! ");
 else
      { for(i=poz;i<n-1;i++)
              x[i]=x[i+1];
              n--;
      }
 printf("\nVectorul este: ");
 tiparire(x,n);
 getch();
}
```

Problema MS4: definiţi 4 vectori cu acelaşi număr e componente, iniţializând rând pe rând trei dintre ei şi calculând cel de-al patrulea după formula:

$$z_i = x_i + y_i - v_i$$

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
const char tt[3]={'x','y','v'};
int z[10],x[10],y[10],v[10];
int w[10][4];
int n;
void main()
  { clrscr();
  printf("\nNr. componente: "); scanf("%d",&n);
  for(int i=0;i<n;i++)
```

```
    { printf(" x[%d]=",i); scanf("%d",&x[i]);
      printf(" y[%d]=",i); scanf("%d",&y[i]);
      printf(" v[%d]=",i); scanf("%d",&v[i]);
      z[i]=x[i]+y[i]-v[i]; };
  printf("\n");
  for(i=0;i<n;i++)
      printf("\n%2d %2d %2d %2d",x[i],y[i],v[i],z[i]); printf("\n\n");
  for(i=0;i<n;i++)
      { for(int j=0;j<3;j++)
          { printf("%c [%d]=",tt[j],i);
            scanf("%d",&w[i][j]); };
        w[i][3]=w[i][0]+w[i][1]-w[i][2]; };
  printf("\n");
  for(i=0;i<n;i++)
      { for(int j=0;j<4;j++)
           printf("%2d ",w[i][j]);
           printf("\n");
      };
  getch();
}
```

Problema MS5: lucrul cu masive influenţează pozitiv performanţele oricărui program. Puneţi în corespondenţă elementele $a_{i,j}$ unui masiv bidimensional, ale cărui valori sunt de forma (i*j), cu elementele unui vector.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int a[4][4];
int b[10];
void main()
{ clrscr();
  for(int i=1;i<4;i++)
    for(int j=1;j<4;j++)
      { b[(i-1)*3+j]=i*j;
        a[i][j]=i*j;
      }
  for(i=1;i<4;i++)
  printf("\n\n\n %d %d ",b[(i-1)*3+i],a[i][i]);
  getch();
}
```

Problema MS6: se consideră o matrice a(5,5) iniţializată cu anumite valori în secţiunea *CONST* şi o altă matrice b(4,4) memorată peste zona de memorie pe care o ocupă matricea a. Să se calculeze suma elementelor de pe diagonala principală a matricei b.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
union zona{
  int a[5][5];
  int b[4][4];
  }z;
int suma(){
   unsigned char i;
   int sum=0,j,k;
   int mat[5][5]={{1,2,3,4,5},{6,7,8,9,10},
```

```
                {11,12,13,14,15},{16,17,18,19,20},{7,1,1,4,2}};
   for(j=0;j<5;j++)
      for(k=0;k<5;k++)
   z.a[j][k]=mat[j][k];
   for(i=0;i<4;i++){
          printf("\n%d ",z.b[i][i]);
          sum+=z.b[i][i];
                   }
   return sum;
          }
void main(){
   int s;
   clrscr();
   s=suma();
   printf("\nSuma elementelor de pe diagonala principala");
   printf("\na matricei b este: %d",s);
   getch();
   }
```

Problema MS7: efectuaţi compunerea matricelor din programul de mai jos ţinând seama de particularităţile de prelucrare.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
const int mat1[3][3]={ { 1,2,3 },
                       { 4,5,6 },
                       { 7,8,9 } };
const int mat2[3][3]={ { -1,-2,-3 },
                       { -4,-5,-6 },
                       { -7,-8,-9 } };
const int mat3[3][3]={ { 1,1,1 },
                       { 2,2,2 },
                       { 3,3,3 } };
int s1,s2,s3;
void main()
{ clrscr();
  s1=s2=s3=0;
  for(int i=0;i<3;i++)
    for(int j=0;j<3;j++)
       { s1+=mat1[i][j];
         s2+=mat2[i][j];
         s3+=mat3[i][j];
       }
 printf("\ns1=%d \ns2=%d \ns3=%d ",s1,s2,s3);
 getch();
 }
```

Problema MS8: indicaţi ce afişează programul:

```
#include<stdio.h>
#include<conio.h>
int i,j,k;
union zona{
    int a[3][3][3];
    int b[9];
    }z;
void main(){
  clrscr();
  for(i=0;i<3;i++)
```

```
    for(j=0;j<3;j++)
      for(k=0;k<3;k++)
         z.a[i][j][k]=-1;
  for(i=0;i<9;i++)
    z.b[i]+=(i+1)*(i+1);
  puts("\n");
  for(i=0;i<9;i++)
    printf(" %d ",z.b[i]);
    puts("\n");
  for(i=0;i<3;i++){
      printf("\n");
      for(j=0;j<3;j++){
         printf("\n");
         for(k=0;k<3;k++)
               printf(" %d ",z.a[i][j][k]);
                     }
                 }
  getch();
}
```

Rezolvare:
În urma execuţiei programul afişează:

0 3 8 15 24 35 48 63 80
0 3 8
15 24 35
48 63 80
-1 -1 -1
-1 -1 -1
-1 -1 -1
-1 -1 -1
-1 -1 -1
-1 -1 -1

Problema MS9: populaţia unei colectivităţi este caracterizată prin 3 grupe de vârstă, 3 categorii de profesiuni şi 3 categorii de salarizare. Astfel, $a_{i,j,k}$ reprezintă numărul de indivizi de grupă de vârsta i, având categoria de profesie j şi salariul de categoria k.

Scrieţi programul care calculează:

- numărul total de indivizi din colectivitate;
- numărul total de indivizi pe grupa de vârstă;
- numărul de indivizi din fiecare grupă de vârstă si categorie de profesie.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
const int a[3][3][3]={ { {1,2,3},{4,5,6},{7,8,9} },
                       { {0,0,0},{1,2,1},{2,2,2} },
                     { {3,2,4},{1,2,3},{5,4,3} } };
int x=0;
int xx[3];;
int xxx[3][3];
void main()
 { clrscr();
   for(int i=0;i<3;i++)
```

```
    { xx[i]=0;
      for(int j=0;j<3;j++)
        { xxx[i][j]=0;
          for(int k=0;k<3;k++)
             { x+=a[i][j][k];
               xx[i]+=a[i][j][k];
               xxx[i][j]+=a[i][j][k];
             }
          printf(" %5d ",xxx[i][j]);
        }
      printf(" xx[%d] %5d \n ",i,xx[i]);
    }
  printf("\nSuma tuturor elem. este: %d ",x);
  getch();
}
```

Problema MS10: să se definească un vector de vectori, să se iniţializeze şi să se însumeze componentele.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int a[4];
typedef a b[3];
const b c={ {1,2,3,4},{5,6,7,8},{6,4,3,2} };
int s=0;
void main()
{ clrscr();
  for(int i=0;i<3;i++)
    for(int j=0;j<4;j++)
       s+=c[i][j];
  printf("\nSuma=%d ",s);
  getch();
}
```

Problema MS11: să se scrie programul care memorează într-o structură de date adecvată elementele unei matrice triunghiulare, cu elementele nenule situate deasupra diagonalei principale.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int mat[5][5];
const mat a={ { 1, 2, 3, 4, 5 },
              { 0, 6, 7, 8, 9 },
              { 0, 0, 10, 11, 12 },
              { 0, 0, 0, 13, 14 },
              { 0, 0, 0, 0, 15 } };
int b[15];
int k=0;
void main()
{ clrscr();
  for(int i=0;i<5;i++)
     for(int j=i;j<5;j++)
        { k++;
          b[k]=a[i][j];
        }
  for(i=0;i<15;i++)
  printf(" %2d ",b[i]);
```

```
  getch();
}
```

Problema MS12: se dă un vector B de 15 elemente de tip întreg. Să se creeze matricea triunghiulară A[5][5] cu elementele nenule deasupra diagonalei principale. Să se adune matricea A cu matricea transpusă a acesteia.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int vec[15];
typedef int mat[5][5];
vec b={ 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 };
int i,j,k=0;
mat a,c,d;
void creare_matrice(vec z,mat x)
{ for(i=0;i<5;i++)
    for(j=0;j<5;j++)
       if (i<=j)
          x[i][j]=z[k++];
       else
           x[i][j]=0;
}
void transp(mat x,mat y)
{ for(i=0;i<5;i++)
     for(j=0;j<5;j++)
         y[j][i]=x[i][j];
}
void tipar(mat x)
{ for(i=0;i<5;i++)
    { printf("\n");
      for(j=0;j<5;j++)
      printf("%2d ",x[i][j]);
    }
}
Void suma(mat x,mat y,mat w)
{ for(i=0;i<5;i++)
    for(j=0;j<5;j++)
       w[i][j]=x[i][j]+y[i][j];
}
void main()
{ clrscr();
  creare_matrice(b,a);
  printf("\n\nMatricea a este: ");
  tipar(a);
  transp(a,c);
  printf("\n\nMatricea b este: ");
  tipar(c);
  suma(a,c,d);
  printf("\n\nMatricea c este: ");
  tipar(d);
  getch();
}
```

Problema MS13: să se formeze matricea triunghiulară A de dimensiune (5,5) cu elementele nenule sub diagonala principală, cu valori citite dintr-un vector B de 15 elemente.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int vec[15];
typedef int mat[5][5];
const vec b={ 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 };
int k=0;
mat a;
void main()
{ clrscr();
  for(int i=0;i<5;i++)
    for(int j=0;j<5;j++)
        If (i>=j)
             a[i][j]=b[k++];
        else
             a[i][j]=0;
    for(i=0;i<5;i++)
         { printf("\n");
           for(int j=0;j<5;j++)
           printf("%2d ",a[i][j]);
         }
  getch();}
```

Problema MS14: să se verifice dacă o matrice a(3,3)de tip întreg, este sau nu o matrice simetrică.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int mat[3][3];
int vb;
mat a;
int simetric(mat a)
{ int vb=1,test;
  for(int i=0;i<3;i++)
    for(int j=2;j>i-1;j--)
      { test=0;
        while ( (test==0)&&(vb) )
        if ( a[i][j]!=a[j][i] ) vb=0;
        else test=1;
      }
  return vb; }
void main()
{ clrscr();
 for(int i=0;i<3;i++)
    for(int j=0;j<3;j++)
      { printf("a(%d)(%d)=",i,j);
        scanf("%d",&a[i][j]);
      };
 vb=simetric(a);
 if (!vb)
     printf("\nMatricea nu este simetrica! ");
 Else
     printf("\nMatricea este simetrica! ");
 getch();}
```

Articolul – structură de date neomogenă şi contiguă

Problema AT1: explicaţi modul de definire, iniţializare şi referire a unei structuri de date de tip articol.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct adresa{
                char strada[40];
                char oras[15];
                char telefon[8];
        };
typedef struct persoana{
              char nume[30];
              struct adresa locul_nasterii;
              unsigned char varsta;
              unsigned int salariu;
              unsigned char vechime;
              struct adresa domiciliul;
              char nr_copii;
         };
persoana muncitor;
void main(){
  strcpy(muncitor.nume,"Popescu Traian");
  strcpy(muncitor.locul_nasterii.strada,"Libertatea Nr. 5");
  strcpy(muncitor.locul_nasterii.oras,"Galati");
  printf("oras.........%s",muncitor.locul_nasterii.oras);
  getch();
}
```

Problema AT2: indicaţi ce afişează programul de mai jos:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct str1{
            char a1[11];
            char a2[11];
                   };
typedef struct str2{
               char b1[22];
                   };
union zona{
     str1 x;
     str2 y;
          }z;
void main(){
  clrscr();
  strcpy(z.x.a1,"aaaaaaaaaaa");
  strcpy(z.x.a2,"bbbbbbbbbbb");
  puts("\n");
  for(int i=0;i<20;i++) printf("%c",z.y.b1[i]);
  printf("%10cde ce oare?",' ');
  getch();}
```

Rezolvare:

În urma execuţiei, programul afişează:

aaaaaaaaaaabbbbbbbbb de ce oare?

Problema AT3: definiţi şi iniţializaţi o matrice a structurii de matrice.

Rezolvare:

În programul alăturat se defineşte masivul bidimensional $art_{i,j}$, ale cărui elemente sunt date de tip articol. Articolul are la rândul său printre membrii un masiv bidimensional $d_{i,j}$. Referirea unui element cu coordonatele k şi j al variabilei art se realizează prin evaluarea expresiei: $Art_{i,j}.d_{k,j}$

```
#include<stdio.h>
#include<conio.h>
typedef struct articol{
                char a[20];
                int b;
                char c;
                short int d[5][5];
                       };
articol art[4][4];
void main(){
  clrscr();
  for(int i=0;i<4;i++)
    for(int j=0;j<4;j++)
      for(int k=0;k<5;k++)
         for(int l=0;l<5;l++)
              art[i][j].d[k][l]=i+j+k+l;
  printf("\nValoarea ultimului element este: %d ",art[3][3].d[4][4]);
  getch();
 }
```

Problema AT4: documentele care însoţesc materialele conţin câmpurile: codul, denumirea, unitatea de măsură şi de la caz la caz cantitatea şi preţul, fie cantitatea existentă în stoc, intrări şi ieşiri, fie numai valoarea materialului. Scrieţi programul care defineşte o structură de tip articol variabil şi realizează iniţializarea acestuia.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
struct capr{
        int cant;
        int pret;};
struct sies{
        int stoc_init;
        int intrari;
        int iesiri;};
union alfa{
       capr cp;
       sies io;
       int val;};
typedef struct {
      int cod,alf;
      char den[30],um[5];
```

```
      union alfa ala;
               }recvar;
recvar articol;
void main(){
  clrscr();
  printf("\ncod material: ");
  scanf("%d",&articol.cod);
  printf("\ndenumire material: ");
  fflush(stdin);
  gets(articol.den);
  printf("\nunitatea de masura: ");
  gets(articol.um);
  printf("\nintroduceti tipul articolului(1,2,3): ");
  scanf("%d",&articol.alf);
  switch(articol.alf){
    case 1:
       printf("\narticol 1:\n");
       printf("\ncantitate:");
       scanf("%d",&articol.ala.cp.cant);
       printf("\npret:");
       scanf("%d",&articol.ala.cp.cant); break;
    case 2:
       printf("\narticol 2:\n");
       printf("\nstoc initial:");
       scanf("%d",&articol.ala.io.stoc_init);
       printf("\nintrari:");
       scanf("%d",&articol.ala.io.intrari);
       printf("\niesiri:");
       scanf("%d",&articol.ala.io.iesiri); break;
    case 3:
       printf("\narticol 3:\n");
       printf("\nvaloare:");
       scanf("%d",&articol.ala.val);break;
       default: printf("\ntip articol necunoscut! ");
  };
  getch();}
```

Problema AT5: definiţi o matrice a[3][3], un vector b[9] şi o structură de tip articol c, având ca membru un vector cu 9 componente. Dacă b[i]=i*i şi &a[1[1]=&b[1]=&c.d[1], să se scrie programul care afişează elementele a[i][j] şi termenii c.d[i] ai structurii de tip articol.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef struct{
           int d[9];
              }c;
union zona{
       int a[3][3];
       int b[9];
       c e;
          }z;
void main(){
  clrscr();
  for(int i=0;i<9;i++)
       z.b[i]=(i+1)*(i+1);
  printf("\nElementele matricei a sunt: ");
  for(i=0;i<3;i++){
            printf("\n");
```

```
            for(int j=0;j<3;j++)
                    printf(" %d ",z.a[i][j]);
             }
  printf("\nElementele articolului sunt: \n");
  for(i=0;i<9;i++)printf(" %d ",z.e.d[i]);
  getch();
 }
```

Problema AT6: scrieţi programul care realizează iniţializarea şi referirea unei structuri de date de tip uniune. Explicaţi rezultatele programului în urma execuţiei.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct union1{
                 char x[10];
                      };
typedef struct union2{
                char y[5];
                      };
typedef struct union3{
                char z[15];
                      };
union zona{
      union1 un1;
      union2 un2;
      union3 un3;
          }zo;
void main(){
  clrscr();
  strcpy(zo.un1.x,"1234567890");
  printf("\n%s",zo.un1.x);
  strcpy(zo.un2.y,"abcde");
  printf("\n%s",zo.un2.y);
  strcpy(zo.un3.z,"123abc456xxx");
  printf("\n%s",zo.un1.x);
  strcpy(zo.un1.x,"1234567890");
  printf("\n%s %s %s",zo.un1.x,zo.un2.y,zo.un3.z);
  getch();
}
```

Problema AT7: despre fiecare student dintr-o grupă se cunosc numele acestuia, notele obţinute la seminar, numărul de prezenţe la seminar. Să se memoreze într-o structură de date corespunzătoare toate aceste informaţii şi să se scrie funcţiile şi procedurile pentru obţinerea următoarelor situaţii:

- afişarea studenţilor în ordine alfabetică;
- afişarea informaţiilor despre un anumit student;
- înscrierea unui nou student în grupă;
- ştergerea unui student din grupă.

Rezolvare:

Structura de date adecvată pentru memorarea informaţiilor despre toţi studenţii dintr-o grupă este vectorul de articole. Se descrie mai întâi structura unui articol, după care se defineşte un vector de articol de dimensiune egală cu numărul maxim de studenţi dintr-o grupă.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct{
        char nume[20];
        int notas[5];
        unsigned char prez;
             }student;
student a[20];
unsigned char sir[20],r;
int i,j,n;
int creare(student a[20]){
  //returneaza nr de studenti introdusi
  int n;
  printf("Numarul de studenti: ");
  scanf("%d",&n);
  for(i=0;i<n;i++){
         printf("\nNumele si prenumele: ");
         fflush(stdin);
         gets(a[i].nume);
         printf("\nnote la seminarii<=10: ");
         for(j=0;j<5;j++) scanf("%d",&a[i].notas[j]);
         printf("\nprezenta: ");
         scanf("%d",&a[i].prez);
                }
  return n;
 }
void sortnume(student a[20],int n){
  //sorteaza studentii în ordine alfabetica
  student s;
  for(i=0;i<n-1;i++){
      for(j=i+1;j<n;j++)
         if((strcmpi(a[i].nume,a[j].nume))>0){
                                             s=a[i];
                                             a[i]=a[j];
                                             a[j]=s;
} } }
void listare(student a[20],int n){
  puts("\n");
  for(i=0;i<n;i++){
      printf("\n%s, note: ",a[i].nume);
      for(j=0;j<5;j++)
          printf(" %d ",a[i].notas[j]);
      printf(", prezenta: %d ",a[i].prez);
                 }
  puts("\n"); }
  int pozitie(student a[20],char nume[20]){
  char vb=1;
  i=-1;
  do{ i++;
    if(!(strcmpi(a[i].nume,nume))) vb=0;
  }while(vb);
  return i;}
int gasit(student a[20],int n,char nume[20]){
  if (!n){printf("\nnume= %s!",nume);return 0;}
  else {
      for(i=0;i<n;i++)
          if(!(strcmpi(a[i].nume,nume))) return 1;
                                         return 0;
       }
```

```
 }
void cauta(student a[20],int n){
  char nume[20];
  printf("\nDati numele studentului! ");
  fflush(stdin);
  gets(nume);
  if(gasit(a,n,nume)){
       i=pozitie(a,nume);
       printf("\nNume si prenume: %s ",a[i].nume);
       printf("\nNote la seminar:\n");
       for(j=0;j<5;j++)
           printf(" %d ",a[i].notas[j]);
       printf("\nPrezenta= %d ",a[i].prez);
                    }
  else printf("\nStudentul nu se afla in aceasta grupa! ");
}
void inserare(student a[20], int *pn){
  student s;
  int poz;
  if(*pn>20)printf("\ngrupa are deja 20 de studenti!");
  else {
     poz=-1;
     printf("\ndati numele studentului ce trebuie inserat: ");
     fflush(stdin);
     gets(s.nume);
     printf("\ndati notele de la seminar: ");
     for(i=0;i<5;i++) scanf("%d",&s.notas[i]);
     printf("\ndati prezenta: ");
     scanf("%d",&s.prez);
     if((strcmpi(s.nume,a[(*pn)-1].nume))>0){
                      a[(*pn)]=s;
                      (*pn)++;
                                             }
        else{
           if(!(*pn)) a[(*pn)++]=s;
           else {if((strcmpi(a[0].nume,s.nume))>0) poz=0;
                 else{poz++;
                for(i=0;i<((*pn)-1);i++)
if(((strcmpi(a[i].nume,s.nume))<0)&&((strcmpi(a[i+1].nume,s.nume))>0))
;
                     poz=i+1; }
           if(poz!=-1){
             for(j=(*pn);j>poz;j--) a[j]=a[j-1];
             a[poz]=s;
             (*pn)++;}
               }
           }
     }
 }
void stergere(student a[20],int *pm){
  char nume[20];
  int k,i;
  if(!(*pm))printf("\nNu este nici un student!");
  else{
   printf("\nDati numele studentului pe care doriti sa-l stergeti! ");
   fflush(stdin);
   gets(nume);
   if (gasit(a,*pm,nume)){
        k=pozitie(a,nume);
        for(i=k;i<(*pm)-1;i++)
            a[i]=a[i+1];
```

```
        (*pm)--; }
   else printf("\nstudentul nu se afla in lista!");
  } }
void meniu(){
  puts("\n");
  printf("\n0 -terminaere");
  printf("\n1 -listare");
  printf("\n2 -cautare");
  printf("\n3 -inserare");
  printf("\n4 -stergere");
 }
void main(){
  int m;
  clrscr();
  m=creare(a);
  sortnume(a,m);
  meniu();
  printf("\nDati optiunea: ");
  fflush(stdin);
  scanf("%c",&r);
  do{
    switch(r){
      case '1':listare(a,m);break;
      case '2':cauta(a,m);break;
      case '3':inserare(a,&m);break;
      case '4':stergere(a,&m);break;
      default:break;
            };
    printf("\nDati optiunea: ");
    fflush(stdin);
    scanf("%c",&r);
   }while(r!='0');
}
```

Variabile pointer

Problema VP1: indicaţi ce realizează programul:

```
#include<stdio.h>
#include<conio.h>
int *pc,**ppc;
int c=7;
void main(){
  clrscr();
  pc=&c;
  ppc=&pc;
  printf("\n%d %d %d",c,*pc,**ppc);
  getch();
}
```

Rezolvare:
În urma execuţiei programul afişează:

Problema VP2: definiţi tipurile de variabile necesare implementării a cinci nivele de referire şi referiţi variabila pointer a ultimului nivel.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int a;
int *pa;
int **ppa;
int ***pppa;
int ****ppppa;
int *****pppppa;
void main(){
  clrscr();
  a=33;
  pa=&a;
  ppa=&pa;
  pppa=&ppa;
  ppppa=&pppa;
  pppppa=&ppppa;
  printf("\n%d",*****pppppa);
  getch();
 }
```

Problema VP3: exemplificaţi definirea şi iniţializarea variabilelor spre pointer şi respectiv pointer spre pointer spre pointer spre întreg.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
int intreg=22;
int *p_int;//pointer spre intreg
int **pp_int;//pointer spre pointer
void main(){
  clrscr();
  p_int=(int *)malloc(sizeof(int));
  p_int=&intreg;
  pp_int=(int **)malloc(sizeof(int*));
  pp_int=&p_int;
  printf("%d",**pp_int);
  getch();
}
```

Problema VP4: să se scrie un program care utilizează pointeri spre pointeri spre pointeri.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
int ***ppp_int;
int **pp_int;
int *p_int;
int intreg=44;
void main(){
  clrscr();
  p_int=(int *)malloc(sizeof(int));
  pp_int=(int **)malloc(sizeof(int*));
```

```
  ppp_int=(int ***)malloc(sizeof(int**));
  p_int=&intreg; pp_int=&p_int; ppp_int=&pp_int;
  printf("%d",***ppp_int);
  getch();
}
```

Problema VP5: indicaţi ce va afişa programul în urma execuţiei:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct articol{
                int camp1;
                char camp2[80];
                unsigned char camp3;
                float camp4;
                       };
int *p_int;
int **p_mat;
articol *p_art;
int mat[3][3]={{1,2,3},{4,5,6},{7,8,9}};
articol art[1]={{100,"sir1",'a',12.3},};
void main(){
  clrscr();
  p_int=(int *)malloc(sizeof(int));
  *p_int=10;
  p_mat=(int**)malloc(3*sizeof(int*));
  for(int i=0;i<3;i++)
        p_mat[i]=(int*)malloc(sizeof(int));
  for(i=0;i<3;i++)
    for(int j=0;j<3;j++)
         p_mat[i][j]=mat[i][j];
  p_art=(articol *)malloc(sizeof(articol));
  p_art=art;
  printf("\nintregul este:...%d",*p_int);
  printf("\nmat[1][1] este:...%d",p_mat[1][1]);
  printf("\ncamp 3 este:...%c",p_art->camp3);
  getch();
}
```

Rezolvare:
În urma execuţiei programul afişează:

intregul este … 10
mat [1][1] este … 5
camp 3 este … a

Problema VP6: definiţi tipuri de date pentru masive uni şi bidimensionale, efectuaţi alocare dinamică, iniţializaţi şi însumaţi elementele şi imprimaţi sumele.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
int **pa;
int *pb;
int s1,s2;
```

```
void main(){
  clrscr();
  pb=(int *)malloc(sizeof(int));
  pa=(int**)malloc(3*sizeof(int*));
  for(int i=0;i<3;i++)
    pa[i]=(int*)malloc(sizeof(int));
  for(i=0;i<10;i++)
    pb[i]=i*i;
  for(i=0;i<3;i++)
    for(int j=0;j<3;j++)
       pa[i][j]=i;
  s1=s2=0;
  for(i=0;i<10;i++)
    S1+=pb[i];
  for(i=0;i<3;i++)
    for(int j=0;j<3;j++)
      s2+=pa[i][j];
  printf("\ns1= %d s2=%d",s1,s2);
  getch();
}
```

Problema VP7: se definesc variabile de tip matrice, vector, dată elementară şi pointerii spre fiecare dintre ele. După iniţializările adecvate interschimbaţi adresele.

Rezolvare:

Se vor defini câte două variabile pointer care se utilizează pentru interschimb.

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct {
            int c1;
               }c;
int a[1]={10},*pa;
int b[1][1]={{20},},(*pb)[1][1];
c cc,*pc;
int d=40,*pd;
int *ppp,*pppp;
int hhh,hhhh;
void main(){
  cc.c1=30;
  clrscr();
  pa=&a[0];
  pb=&b;
  pc=&cc;
  pd=&d;
  ppp=pd;
  pppp=pa;
  pd=pppp;
  pa=ppp;
  printf("\n %d %d %d %d",a[0],b[0][0],cc.c1,d);
  hhh=pc->c1;
  hhhh=*pb[0][0];
  *pb[0][0]=hhh;
  pc->c1=hhhh;
  printf("\n %d %d %d %d",pa[0],(*pb)[0][0],pc->c1,*pd);
  getch();
}
```

Problema VP8: explicaţi ce afişează programul de mai jos:

```
#include<stdio.h>
#include<conio.h>
typedef struct{
            int * b;
            int c;
              }a;
a aa,*pa;
void main(){
  clrscr();
  aa.c=7;
  aa.b=&aa.c;
  printf("\naa.c = %d *aa.b = %d",aa.c,*aa.b);
  pa=&aa;
  printf("\n*pa->b = %d",*pa->b);
  getch();
}
```

Rezolvare:
Se va afişa:

*aa.c = 7 *aa.b = 7*
**pa->b = 7*

Problema VP9: explicaţi ce afişează programul de mai jos:

```
#include<stdio.h>
#include<conio.h>
typedef struct {
            int c[2];
            int *d[2];//vector de pointeri
               }b;
b bb[2]; //vector de articole
b *pbc[2];//vector de pointeri la articole
b (*ppp)[2];//pointer la vector de articole
void main(){
  clrscr();
  bb[0].c[0]=11;
  bb[0].d[0]=&bb[0].c[0];
  bb[0].c[1]=22;
  bb[0].d[1]=&bb[0].c[1];
  bb[1].c[0]=11;
  bb[1].d[0]=&bb[1].c[0];
  pbc[0]=&bb[0];
  printf("\n *pbc[0]->d[1] = %d",*pbc[0]->d[1]);
  ppp=&bb;
  printf("\n *ppp[0]->d[1] = %d",*ppp[0]->d[1]);
  getch();
}
```

Rezolvare:
Se va afişa:

**pbc[0]->d[1] = 22*
**ppp[0]->d[1] = 22*

Problema VP10: se consideră variabilele elementare a, b, c şi d şi un vector m cu patru elemente. Scrieţi programul care însumează elementele a, b, c, d în s1 şi elementele vectorului m în s2, în cadrul aceleaşi structuri repetitive. Să se pună în evidenţă comutativiatea atributelor vector de pointeri şi pointeri spre vectori.

Rezolvare:

Se defineşte un vector de pointeri spre întreg numit x, care se iniţializează cu adresele lui a, b, c şi d. Se evidenţiază definirea pointerului spre un vector pm.

```
#include<stdio.h>
#include<conio.h>
int a=7,b=10,c=13,d=20;
int m[4]={7,10,13,20};
int (*pm)[4];//un pointer la vector
int *x[4]; //un vector de pointeri
int i,s1,s2;
void main(){
  x[0]=&a;
  x[1]=&b;
  x[2]=&c;
  x[3]=&d;
  pm=&m;
  s1=s2=0;
  for(i=0;i<4;i++){
      s1+=*x[i];
      s2+=(*pm)[i]; }
  printf("\ns1= %d s2=%d",s1,s2);
  getch(); }
```

Problema VP11: se consideră patru numere memorate în zone asociate unor variabile elementare. Scrieţi programul care permite tratarea acestora ca elemente ale unui masiv bidimensional.

Rezolvare:

Se defineşte o matrice a[2][2] care se iniţializează cu adresele variabilelor elementare dispersate ca poziţionare.

```
#include<stdio.h>
#include<conio.h>
int *a[2][2];//matrice de pointeri
int b=1,c=13,d=172,e=3333;
void main(){
  clrscr();
  a[0][0]=&b;
  a[1][0]=&d;
  a[0][1]=&e;
  a[1][1]=&c;
  for(int i=0;i<2;i++)
     for(int j=0;j<2;j++)
         printf("\na[%d][%d]= %d",i+1,j+1,*a[i][j]);
  getch();
}
```

Problema VP12: explicaţi ce afişează programul:

```
#include<stdio.h>
#include<conio.h>
unsigned char i;
union pzona{
      int (*pa)[5][5];//pointer la matrice
      int (*pb)[4][4]; }z;
int *pp;
int a[5][5]={{1,2,3,4,5},
            {6,7,8,9,10},
            {11,12,13,14,15},
            {16,17,18,19,20},
            {7,1,1,4,2}};
int suma(int(*pb)[4][4],unsigned char n){
  unsigned char i;
  int sum=0;
  for(i=0;i<n;i++){
      printf(" %d ",(*pb)[i][i]);
              sum+=(*pb)[i][i];
        }
  return sum;
}
void main(){
  clrscr();
  z.pa=&a;
  int s=0;
  for(i=0;i<5;i++){
     s+=(*z.pa)[i][i];
     printf(" %d ",(*z.pa)[i][i]);
                  }
  printf("\nSuma pentru a[1..5][1..5]= %d\n\n",s);
  printf("\nSuma pentru a[1..5][1..5]= %d\n\n",suma(z.pb,5));
  printf("\nSuma pentru a[1..4][1..4]= %d\n\n",suma(z.pb,4));
  getch();
}
```

Rezolvare:
Programul afişează:

1 7 13 19 2
Suma pentru a[1..5][1..5] = 42
1 6 11 16 7
Suma pentru a[1..5][1..5] = 41
1 6 11 16
Suma pentru a[1..4][1..4] = 34

Problema VP13: afişaţi elementele unui vector de tip întreg, accesând elementele acestuia prin intermediul unui vector de pointeri spre întreg şi a unui pointer spre vectori de pointeri spre întreg.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int a[5]={1,2,3,4,5};
int *p[5];//vector de pointeri
int *(*pp)[5];
void main(){
```

```
  clrscr();
  for(unsigned char i=0;i<5;i++)
  p[i]=&a[i];
  pp=&p;
  for(i=0;i<5;i++)
      printf("\n %d %d", *(*pp)[i],*p[i]);
  getch();
}
```

Problema VP14 Să se calculeze sumele a trei matrice de aceeaşi dimensiune, accesând elementele matricelor prin intermediul unui vector de pointeri spre matrice.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int (*vp[3])[2][3];
unsigned char i,j,k;
int s[3]={0,0,0};
int a[2][3]={{1,1,1},{2,2,2}};
int b[2][3]={{3,3,3},{4,4,4}};
int c[2][3]={{5,5,5},{6,6,6}};
void main(){
  clrscr();
  unsigned char var='a';
  vp[0]=&a;
  vp[1]=&b;
  vp[2]=&c;
  for(k=0;k<3;k++){
      for(i=0;i<2;i++)
          for(j=0;j<3;j++)
             s[k]+=(*vp[k])[i][j];
       printf("\nSuma matricei %c este: %d ",var,s[k]);
       var++;
                    }
  getch();
}
```

Problema VP15: să se calculeze sumele a trei matrice de aceeaşi dimensiune, accesând elementele matricelor prin intermediul unui vector de pointeri spre pointeri spre matrice.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int (*(*vpp[3]))[2][3];
int (*vp[3])[2][3];
unsigned char i,j,k;
int s[3]={0,0,0};
int a[2][3]={{1,1,1},{2,2,2}};
int b[2][3]={{3,3,3},{4,4,4}};
int c[2][3]={{5,5,5},{6,6,6}};
void main(){
  clrscr();
  unsigned char var='a';
  vp[0]=&a;
  vp[1]=&b;
  vp[2]=&c;
  for(i=0;i<3;i++)
```

```
  vpp[i]=&vp[i];
  for(k=0;k<3;k++){
      for(i=0;i<2;i++)
          for(j=0;j<3;j++)
               s[k]+=(*(*vpp[k]))[i][j];
      printf("\nSuma matricei %c este: %d ",var,s[k]);
      var++;
                }
  getch();
}
```

Problema VP16: să se calculeze sumele a trei matrice de aceeaşi dimensiune, accesând elementele matricelor prin intermediul unui pointer spre un vector de pointeri spre pointeri spre matrice.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int (*(*(*pvpp)[3]))[2][3];
int (*(*vpp[3]))[2][3];
int (*vp[3])[2][3];
unsigned char i,j,k;
int s[3]={0,0,0};
int a[2][3]={{1,1,1},{2,2,2}};
int b[2][3]={{3,3,3},{4,4,4}};
int c[2][3]={{5,5,5},{6,6,6}};
void main(){
  clrscr();
  unsigned char var='a';
  vp[0]=&a;
  vp[1]=&b;
  vp[2]=&c;
  for(i=0;i<3;i++)
    vpp[i]=&vp[i];
  pvpp=&vpp;
  for(k=0;k<3;k++){
       for(i=0;i<2;i++)
          for(j=0;j<3;j++)
             s[k]+=(*(*(*pvpp)[k]))[i][j];
       printf("\nSuma matricei %c este: %d ",var,s[k]);
       var++;
                 }
  getch();
}
```

Problema VP17: să se însumeze separat elementele a trei vectori folosind o singură instrucţiune în care rând pe rând vor fi referite elementele acestora.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int x[10]={1,2,3,4,5,6,7,8,9,10};
int y[10]={10,9,8,7,6,5,4,3,2,1};
int z[10]={1,1,1,1,1,1,1,1,1,1};
int s[3]={0,0,0};
int (*pv[3])[10];//vector de pointeri la vectori
void main(){
  clrscr();
```

```
  pv[0]=&x;
  for(int k=0;k<3;k++){
     for(int i=0;i<10;i++)
       s[k]+=(*pv[k])[i];
         printf("\n %d",s[k]);
         pv[k+1]=++pv[k];
          }
  getch();
}
```

Problema VP18: scrieţi programul care efectuează însumarea elementelor unui vector x de n componente cu instrucţiunile:

- $s1+=x_i$;
- $s2 +=(*px1)_i$; $s3+=(*px2)_i$.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
int s1,s2,s3;
int (*px1)[10];
int *px2[10];
int x[10]={1,2,3,4,5,6,7,8,9,10};
void main(){
  clrscr();
  s1=s2=s3=0;
  px1=&x;
  for(int i=0;i<10;i++)
  px2[i]=&x[i];
  for(i=0;i<10;i++){
    s1+=x[i];
    s2+=(*px1)[i];
    s3+=*px2[i];
        }
  printf("\n %d %d %d",s1,s2,s3);
  getch();
}
```

Problema VP19: explicaţi ce afişează programul:

```
#include<conio.h>
#include<stdio.h>
int a1[3][3]={{1,2,3},{4,5,6},{7,6,1}};
int a2[3][3]={{1,1,2},{3,3,3},{4,4,1}};
int a3[3][3]={{1,1,2},{4,4,7},{3,3,4}};
int a4[3][3]={{4,4,4},{2,1,1},{4,3,2}};
int a5[3][3]={{1,0,0},{0,0,7},{2,3,4}};
int a6[3][3]={{3,3,3},{3,3,1},{2,7,6}};
int a7[3][3]={{1,0,0},{6,7,8},{3,6,9}};
int (*pa[7])[3][3];
int (*pb[7])[3][3];
int (*ppb)[3][3];
int b1[3][3],b2[3][3],b3[3][3],b4[3][3],b5[3][3],b6[3][3],b7[3][3];
int (*calcul(int (*px)[3][3],int (*pxx)[3][3]))[3][3]{
int ii,j;
for(ii=0;ii<3;ii++){
   printf("\n");
   for(j=0;j<3;j++){
      (*pxx)[ii][j]=( ((*px)[ii][j]) * ((*px)[ii][j]) );
      printf(" %d ",(*pxx)[ii][j]);
```

```
                    }
           }
return pxx;
}
void main(){
  clrscr();
  pa[0]=&a1;
  pa[1]=&a2;
  pa[2]=&a3;
  pa[3]=&a4;
  pa[4]=&a5;
  pa[5]=&a6;
  pa[6]=&a7;
  pb[0]=&b1;
  pb[1]=&b2;
  pb[2]=&b3;
  pb[3]=&b4;
  pb[4]=&b5;
  pb[5]=&b6;
  pb[6]=&b7;
  for(int i=0;i<7;i++){
      pb[i]=calcul(pa[i],pb[i]);
      printf("\n\n");
      for(int k=0;k<3;k++){
        printf("\n");
        for(int j=0;j<3;j++)
           printf(" %d ",(*pb[i])[k][j]);
        printf("\n");
                            }
                    }
  getch();
}
```

Problema VP20: explicaţi ce afişează programul:

```
#include<conio.h>
#include<stdio.h>
int (*v[3])[3][3];
int (*(*ppp)[3])[3][3];
int(*(*(*ph))[3])[3][3];
int s[3],m[3][3]={{1,2,8},{4,6,7},{6,7,5}};
void main(){
  clrscr();
  for(int i=0;i<3;i++)
      v[i]=&m;
  ppp=&v;
  ph=&ppp;
  for(i=0;i<3;i++){
      s[i]=0;
      for(int j=0;j<3;j++)
          s[i]+=(*(*(*ph))[i])[i][j];
                  }
   for(i=0;i<3;i++)
      printf(" %d ",s[i]);
  getch();}
```

Rezolvare:

În urma execuţiei programul afişează:

Problema VP21: să se calculeze fie suma elementelor unei matrice, fie suma elementelor unui vector alocate dinamic. În funcţie de o anumită opţiune se alocă şi se iniţializează memorie pentru masivul unidimensional sau pentru masivul bidimensional, astfel încât: $a_{i,j}=i*j$; $b_i=i*i$. Se afişează suma elementelor masivului prelucrat şi se eliberează memoria rezervată.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int mat[3][3];
typedef int vect[3];
typedef mat *pmat;
typedef vect *pvect;
pmat pa;
pvect pb;
int k;
int s=0;
void main()
  { printf("\nIntroduceti optiunea(1/0): ");
    scanf("%d",&k);
    if (!k)
       { pb=new vect[3];
         for(int i=0;i<10;i++)
               { *pb[i]=(i+1)*(i+1);
                  s+=*(pb[i]);
               }
         printf("\nsuma vect=%d",s);
         delete pb;
       }
   else
     { pa=new mat[3];
       for(int i=0;i<3;i++)
         for(int j=0;j<3;j++)
             { *pa[i][j]=(i+1)*(j+1);
               s+=*pa[i][j];
             }
       printf("\nsuma matrice=%d",s);
      delete pa;
     }
  getch();
}
```

Problema VP22: indicaţi ce afişează programul:

```
#include<conio.h>
#include<stdio.h>
#include<dos.h>
int*ppc;
int c[5][2]={{1,2},{3,4},{5,6},{7,8},{9,10}};
int b[10]={11,12,13,14,15,16,17,18,19,20};
void main(){
  clrscr();
  printf("\n");
  for(int i=0;i<10;i++){
    ppc=(int*)MK_FP(FP_SEG(c[i]),FP_OFF(c[i]));
  printf(" %d ",*ppc);
   }
getch(); }
```

Rezolvare:

1 3 5 7 9 11 13 15 17 19

Problema VP23: indicaţi ce afişează programul:

```
#include<conio.h>
#include<stdio.h>
#include<dos.h>
int *ppc;
int c[2][5]={{1,2,3,4,5},{6,7,8,9,10}};
int b[10]={11,12,13,14,15,16,17,18,19,20};
void main(){
  clrscr();
  printf("\n");
  for(int i=0;i<4;i++){
     ppc=(int*)MK_FP(FP_SEG(c[i]),FP_OFF(c[i]));
     printf(" %d ",*ppc);
      }
  getch();
}
```

Rezolvare:
Programul în urma execuţiei va afişa:

1 6 11 16

Problema VP24: se consideră masivele unidimensionale b şi c având 10 componente. Exemplificaţi existenţa concatenării folosind variabila FP_SEG, FP_OFF şi lungimea tipului elementelor ce alcătuiesc respectivele masive pentru efectuarea referirii elementelor lor.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
#include<dos.h>
int *ppc;
int c[10]={1,2,3,4,5,6,7,8,9,10} ;
int b[10]={11,12,13,14,15,16,17,18,19,20};
void main(){
  clrscr();
  printf("\n");
  for(int i=0;i<20;i++){
       ppc=(int*)MK_FP(FP_SEG(c+i),FP_OFF(c+i));
       printf("%d ",*ppc);
                        }
  getch();
}
```

Problema VP25: indicaţi ce afişează programul:

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
int x[10]={1,2,3,4,5,6,7,8,9,10};
int *px;
```

```
unsigned int sg,ofsetul;
void main(){
  clrscr();
  int s=0;
  sg=FP_SEG(x);
  ofsetul=FP_OFF(x);
  for(int i=0;i<10;i++){
          px=(int *)MK_FP(sg+2*i,ofsetul+2*i);
          s+=*px;
                        }
  printf(" %d",s);
  getch();
}
```

Rezolvare:
În urma execuţiei, programul va afişa: 55

Problema VP26: indicaţi ce afişează programul:

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
int x[10]={1,2,3,4,5,6,7,8,9,10};
int *c;
unsigned int a,b;
void main(){
  clrscr();
  int s=0;
  a=FP_SEG(x);
  b=FP_OFF(x);
  for(int i=0;i<10;i++){
         c=(int *)MK_FP(a+i*sizeof(int),b+i*sizeof(int));
         s+=*c;
                      }
  printf(" %d",s);
  getch();
}
```

Rezolvare:
Programul în urma execuţiei va afişa: *suma = 55*

Problema VP27: să se scrie programul care apelează o funcţie de calcul a salariului cuvenit unui muncitor. Funcţia returnează un pointer la o zonă de memorie alocată dinamic de aceasta.
Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct art { int marca;
             char nume[20];
             long int nrore;
             long int salorar;
             long int retineri;
             long int sal;
           } ;
typedef struct art *part;
part pp;
```

```
part citire()
{ part p;
  P=new art;
  fflush(stdin);
  printf("marca: "); scanf("%d",&p->marca);
  printf("nume: "); fflush(stdin); gets(p->nume);
  printf("salariu orar: "); scanf("%ld",&p->salorar);
  printf("numar ore: "); scanf("%ld",&p->nrore);
  printf("retineri: "); scanf("%ld",&p->retineri);
  printf("% ld ",(long)((p->nrore)*(p->salorar)) );
  p->sal=(long)((p->nrore)*(p->salorar)-(p->retineri));
  printf("%ld",p->sal);
  return p; }
void main()
{ clrscr();
  pp=citire();
  fflush(stdout);
  printf("\n %s salar: %ld",pp->nume,pp->sal);
  getch();
}
```

Problema VP28: să se scrie programul care determină studenţii dintr-o grupă care au nota cea mai mare la un anumit examen, folosind un pointer la un vector de articole.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
typedef struct str{
                char nume[25];
                int nota; };
str b[20];
str (*pb)[20];
unsigned char i,n,max=1;
void main(){
  clrscr();
  printf("\nDati numarul de studenti din grupa: ");
  scanf("%d",&n);
  pb=&b;
  for(i=0;i<n;i++){
       printf("\nNumele: ");fflush(stdin);
       gets((*pb)[i].nume);
       printf("\nNota: ");fflush(stdin);
       scanf("%d",&(*pb)[i].nota);
       if((*pb)[i].nota>max)
              max=(*pb)[i].nota; }
  for(i=0;i<n;i++)
      if(max==(*pb)[i].nota)
         printf("\n nume: %s nota: %d",(*pb)[i].nume,(*pb)[i].nota);
  getch();
}
```

Problema VP29: exemplificaţi definirea unui vector de structură, iniţializarea componentelor lui şi afişarea membrilor folosind operatorul de referire într-o expresie în care apare un vector de pointeri spre structură.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
```

```
typedef struct articol{
                  int camp1;
                  int camp2;
                     }ar;
ar *p_vect_articol[3];//vector de pointeri spre structura
ar vect_articol[3]={{1,2},{3,4},{5,6},};
void main(){
  clrscr();
  for(int i=0;i<3;i++)
      p_vect_articol[i]=&vect_articol[i];
  printf("\n%d",vect_articol[1].camp2);
  printf("\n%d",p_vect_articol[1]->camp2);
  getch();
}
```

Problema VP30: indicaţi ce afişează programul:

```
#include<stdio.h>
#include<conio.h>
struct str
  { int elem;
    int *pelem;
  };
typedef int mat[1][1];
typedef struct str *pstr;
typedef mat *pmat;
mat a;
str b;
pmat pa;
pstr pb;
pstr initstr(pmat pm)
{ pstr ps;
  ps=&b;
  ps->pelem=&(ps)->elem;
  ps->elem=(*pm)[0][0];
  return ps;
}
void main()
{ clrscr();
  a[0][0]=111;
  pa=&a;
  pb=initstr(pa);
  printf(" ******* %d ******* ",*pb->pelem);
  getch();
}
```

Rezolvare:

******* *111* *******

Problema VP31: indicaţi şi explicaţi rezutatele afişate de programul:

```
#include<stdio.h>
#include<conio.h>
typedef int vec[2];
typedef int *pvec[2];
typedef pvec *ppvec;
struct str
       { vec aa;
```

```
         pvec paa;
         ppvec pppa;
       };
typedef str *pstr;
typedef pstr ps[2];
typedef ps *pps;
str st[2];
ps pst;
pps pss;
ppvec pa;
void main()
{ clrscr();
  st[0].aa[0]=100;
  st[0].aa[1]=200;
  st[0].paa[0]=&st[0].aa[0];
  st[0].paa[1]=&st[0].aa[1];
  st[0].pppa=(ppvec)&st[0].paa[0];
  st[1].aa[0]=300;
  st[1].aa[1]=400;
  st[1].paa[0]=&st[1].aa[0];
  st[1].paa[1]=&st[1].aa[1];
  st[1].pppa=(ppvec)&st[1].paa[0];
  pst[0]=&st[0];
  pst[1]=&st[1];
  pss=(pps)&pst[0];
  for(int i=0;i<2;i++)
    for(int j=0;j<2;j++)
       { printf(" * %d ** %d ",st[i].aa[j],*(*pst[i]).paa[j]);
         printf("++ %d -- ",*((*(*pss)[i]).paa[j]));
         printf(" %d\n", *(*(*(*pss)[i]).pppa)[j]);
       }
  getch();
}
```

Rezolvare:

```
* 100 ** 100 ++ 100 -- 100
* 200 ** 200 ++ 200 -- 200
* 300 ** 300 ++ 300 -- 300
* 400 ** 400 ++ 400 -- 400
```

Problema VP32: indicaţi ce realizează programul:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct a
   { char a1[20];
   } ;
struct b
     { char b1[20];
     } ;
typedef struct a *pa;
typedef struct b *pb;
a x;
b y;
pa px;
pb py;
void *pp;
```

```
void main()
{ clrscr();
  strcpy(x.a1,"abcdefghijklmnoprs");
  strcpy(y.b1,"1234567890");
  pp=&x;
  py=(b*)(pp);
  printf("%s\n",py->b1);
  pp=&y;
  px=(a*)pp;
  printf("%s",px->a1);
  getch();
}
```

Rezolvare:
Programul afişează:

abcdefghijklmnoprs
1234567890

Problema VP33: definiţi o structură formată din doi membrii – primul va conţine variabila A iar al doilea membru adresa lui B. Se alocă dinamic deasemenea o variabilă B care se referă şi se iniţializează. Să se calculeze expresia: A:=A+B

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct str
     { int a;
       int *pb;
     } ;
typedef struct str *pstr;
pstr pa;
void main()
{ clrscr();
  pa=(str *)malloc(sizeof(str));
  pa->a=7;
  pa->pb=(int*)malloc(sizeof(int));
  *pa->pb=8;
  pa->a+=*pa->pb;
  printf("%d",pa->a);
  getch();
}
```

Problema VP34: definiţi structura pe care o iniţializaţi într-un program astfel încât executarea instrucţiunii **printf(''\n %d",***((**(*x).y).z))** să afişeze *-1*.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int a;
typedef a *pa;
typedef pa *ppa;
typedef ppa *pppa;
struct r1
```

```
    { pppa z;
    };
typedef struct r1 *pr1;
typedef pr1 *ppr1;
struct r2
   { ppr1 y;
   };
typedef struct r2 *pr2;
a n;
pa pn;
ppa ppn;
pppa pppn;
struct r1 art1;
pr1 part1;
ppr1 ppart1;
struct r2 art2;
pr2 x;
void main()
{ clrscr();
  n=-1;
  pn=&n;
  ppn=&pn;
  pppn=&ppn;
  art1.z=pppn;
  part1=&art1;
  ppart1=&part1;
  art2.y=ppart1;
  x=&art2;
  printf("\n %d ",***((**(*x).y).z) );
  getch();
}
```

Problema VP35: evaluaţi într-un program expresia **(*(*x)[i]).y[j])
Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int a;
typedef a *pa;
typedef pa *ppa;
typedef ppa v1[2];
struct r1
      { v1 y;
      };
typedef struct r1 *pr1;
typedef pr1 v2[2];
typedef v2 *pv2;
a n1,n2,m1,m2;
pa pn1,pn2,pm1,pm2;
ppa ppn1,ppn2,ppm1,ppm2;
struct r1 art1,art2;
v2 vec;
pv2 x;
void main()
{ clrscr();
  n1=100; n2=101;
  pn1=&n1; pn2=&n2;
  ppn1=&pn1; ppn2=&pn2;
  art1.y[0]=ppn1;
  art1.y[1]=ppn2;
```

```
  m1=200; m2=201;
  pm1=&m1; pm2=&m2;
  ppm1=&pm1; ppm2=&pm2;
  art2.y[0]=ppm1;
  art2.y[1]=ppm2;
  vec[0]=&art1;
  vec[1]=&art2;
  x=&vec;
  for(int i=0;i<2;i++)
     for(int j=0;j<2;j++)
         printf("\n **(*(*x)[i]).y[j]=== %d ",i,j,**(*(*x)[i]).y[j]);
  getch();
}
```

Problema VP36: definiţi şi referiţi un masiv tridimensional punând în corespondenţă adresele elementelor de pe prima linie şi coloana a masivelor bidimensionale în care se descompune, cu elementele unui vector de pointeri. Iniţializarea masivului tridimensional se realizează prin date introdusede la terminal utilizând aceşti pointeri.

Rezolvare:

```
#include<dos.h>
#include<stdio.h>
unsigned int aaa,bbb;
int a[3][3][3];
int *vp[3];
int *pp;
int j,k;
char rasp='1';
void main(){
  clrscr();
  for(int i=0;i<3;i++)
     for(j=1;j<3;j++)
        for(k=0;k<3;k++)
             a[i][k][j]=0;
  for(i=0;i<3;i++)
     vp[i]=&a[i][0][0];
  while(rasp!=' '){
     printf("\nintroduceti coordonatele:\n ");
     printf("\ncoordonata i : ");scanf("%d",&i);i--;
     printf("coordonata j : ");scanf("%d",&j);j--;
     printf("coordonata k : ");scanf("%d",&k);k--;
     aaa=FP_SEG(vp[i]);
     bbb=FP_OFF(vp[i]);
      aaa+=(j*3+k)*2;
      bbb+=(j*3+k)*2;
      pp=(int *)MK_FP(aaa,bbb);
      printf("\ndati elementul: ");
      scanf("%d",pp);
      printf("\nContinuati introducerea? [pentru nu,tastati spatiu]:
");
      fflush(stdin);scanf("%c",&rasp);
    }
  for(i=0;i<3;i++){
     for(j=0;j<3;j++){
       for(k=0;k<3;k++)
          printf(" %d ",a[i][j][k]);
       printf("\n");
                    }
     printf("\n*******\n");
```

```
                }
  getch();
}
```

Problema VP37: definiţi o matrice, iniţializaţi acest nou tip de dată şi referiţi în două moduri distincte elementele sale, în programul C ce alege elementul minim şi identifică poziţia acestuia.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
#include<alloc.h>
int poz[4]={1,1,1,1};
int min,i,j,h,k,d1,d2,d3,d4;
int (*p1)[10][10];
int ((*pp1)[10][10])[10][10];
void main(){
  clrscr();
  printf("\n Dati dimensiunile tabloului:\n");
  printf("d1= ");scanf("%d",&d1);
  printf("d2= ");scanf("%d",&d2);
  printf("d3= ");scanf("%d",&d3);
  printf("d4= ");scanf("%d",&d4);
  for(i=0;i<d1;i++)
     for(j=0;j<d2;j++)
        for(h=0;h<d3;h++)
           for(k=0;k<d4;k++){
              fflush(stdin);
              scanf("%d",&(*p1)[h][k]);
              (*pp1)[i][j][h][k]=(*p1)[h][k];
                              }
  min=((*pp1)[0][0])[0][0];
  for(i=0;i<d1;i++)
    for(j=0;j<d2;j++)
        for(h=0;h<d3;h++)
          for(k=0;k<d4;k++)
                if((*pp1)[i][j][h][k]<min){
                      min=((*pp1)[i][j])[h][k];
                      poz[0]=i+1;
                      poz[1]=j+1;
                      poz[2]=h+1;
                      poz[3]=k+1;
                                          }
  printf("\nElementul minim %d apare prima data in aceasta
structura",min);
  printf("\npe linia %d, col %d, din matricea de pe",poz[2],poz[3]);
  printf(" linia %d coloana %d",poz[0],poz[1]);
  getch();
}
```

Liste

Problema LS1: să se creeze o listă liniară simplu înlănţuită cu trei noduri, să se iniţializeze şi să se afişeze elementele ei.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
```

```
#include<stdlib.h>
struct lista
    { int val;
      lista *urm;
    };
lista *pa,*pb,*pc;
void main()
{ clrscr();
  pa=new(lista);
  pb=new(lista);
  pc=new(lista);
  pa->urm=pb;
  pb->urm=pc;
  pc->urm=NULL;
  pa->val=5;
  pb->val=50;
  pc->val=500;
  printf("\n %d",pa->val);
  printf("\n %d",pa->urm->val);
  printf("\n %d",pa->urm->urm->val);
  getch();
}
```

Problema LS2: definiţi şi iniţializaţi o listă simplu înlănţuită. Efectuaţi ştergerea temporară a unui element. Afişaţi lista iniţială, lista modificată, reactivaţi elementul.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
struct lista
      { int info;
        int vb;
        lista *urm;
      };
lista *cpl;
int r1;
char r2;
int n,elem;
lista* creare(int nr)
  { lista *p1=NULL;
    if (!nr)
       return NULL;
    else
      { nr--;
        p1=new lista;
        printf("Dati informatia: ");
        scanf("%d",&p1->info);
        p1->vb=1;
        p1->urm=creare(nr);
        return p1;
      }
  }
int stergere(lista *p1,int k,int opt)
{ int vbb=0;
  while ( (p1)&&(!vbb) )
      { if (p1->info==k)
            { if (!opt)
                { p1->vb=0;
```

```
                  vbb=1;
                 }
               else
                { p1->vb=1;
                  vbb=1;
                 }
              }
      else
         p1=p1->urm;
       }
  if (vbb)
     return 1;
  else
      return 0;
}
void tipar(lista *p1)
{ while (p1)
      { if (p1->vb)
            printf(" %d ",p1->info);
        p1=p1->urm;
       }
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. ale listei: "); scanf("%d",&n);
  cpl=creare(n);
  tipar(cpl);
  printf("\nElem. pe care doriti sa-l stergeti: "); scanf("%d",&elem);
  r1=stergere(cpl,elem,0);
  if (!r1)
     printf("\nNu exista %d in lista! ",elem);
  else
     { printf("\n%d este sters temporar",elem);
       printf("\nLista este: "); tipar(cpl);
       printf("\nDoriti sa-l rescrieti(y/n)");
       r2=getche();
       if (tolower(r2)=='y')
           { r1=stergere(cpl,elem,1);
                printf("\nLista devine: ");
                tipar(cpl); }
           }
  getch();
}
```

Problema LS3: se consideră o listă ce conţine ca elemente care au ca informaţie utilă: cod produs, cantitate şi preţ. Scrieţi şi apelaţi funcţia care calculează total valoare pentru materialele existente în listă.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
struct prod
          { char codp[5];
            int cant;
            double pret;
            prod *urm;
          };
prod *lista=NULL;
double tot;
```

```
double total(prod *list)
{ double s=0;
   while (list)
        { s+=(double)(list->cant)*(list->pret);
          list=list->urm;
        }
   return s;
}
prod *formeaza(prod *list)
{ char sf='y';
  prod *el;
  double pret;
  while (tolower(sf)=='y')
      { el=new prod;
        printf("\nNou produs.Dati codul= "); fflush(stdin);
        gets(el->codp);
        printf("In cantitate: "); scanf("%d",&el->cant);
        printf("La pretul: ");
        scanf("%lf",&pret); el->pret=pret;
        el->urm=list;
        list=el;
        printf("\nMai sunt produse? [y/n] ");
        sf=getche();
      }
  return list;
}
void main()
{ clrscr();
  lista=formeaza(lista);
  tot=total(lista);
  printf("\nTotalul este: %10.2f",tot);
  getch();
}
```

Problema LS4: să se scrie procedurile care realizează diferite modalităţi de tipărire a unei liste simplu înlănţuite.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct nod
     { int info;
       nod *urm;
     };
nod *cpl,*p,*pp;;
void nerec(nod *pp)
{ while (pp)
    { printf(" %d ",pp->info);
      pp=pp->urm;
    }
}
void rec(nod *pp)
{ if (pp)
    { rec(pp->urm);
      printf(" %d ",pp->info);
     } }
void rec1(nod *pp)
{ if (pp)
     { printf(" %d ",pp->info);
       rec1(pp->urm);
```

```
     }
}
void main()
{ clrscr();
  p=new nod;
  cpl=p;
  p->info=0;
  for(int i=0;i<5;i++)
     { pp=new nod;
       p->urm=pp;
       pp->info=i+1;
       p=pp;
     }
  pp->urm=NULL;
  printf("\nScriere nerecursiva: \n");
  nerec(cpl);
  printf("\nScriere recursiva cap-coada: \n");
  rec1(cpl);
  printf("\nScriere recursiva coada-cap: \n");
  rec(cpl);
  getch();
}
```

Problema LS5: să se scrie programul pentru adunarea a două matrice rare, memorate sub formă de liste liniare simplu înlănţuite.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct mat
       { int lin;
         int col;
         int info;
         mat *urm; };
typedef int matrice[10][10];
typedef matrice *pmatrice;
int n1,contor1,contor2;
int vb1,vb2;
mat *cpla,*cplb,*cplc,*pa,*pb,*pc,*prar;
pmatrice pm1,pm2;
mat *creare(pmatrice pm)
{ int vb=0;
  mat *cpl=NULL,*pa,*prar;
  for(int i=0;i<n1;i++)
     for(int j=0;j<n1;j++)
        if (*pm[i][j])
            { pa=new mat;
              pa->lin=i;
              pa->col=j;
              pa->info=*pm[i][j];
              if (!vb)
                 { cpl=pa;
                   prar=pa;
                   pa->urm=NULL;
                   vb=1;
                 }
             else
                { prar->urm=pa;
                  prar=pa;
                }
```

```
            }
  pa->urm=NULL;
  return cpl;
}
void tipareste(mat *pl)
{ while (pl)
   { printf("\nLinia:%1d Coloana:%1d Info:%d ",pl->lin,pl->col,pl-
>info);
   pl=pl->urm;
   }
}
void copie(mat *px,mat *pc)
{ mat *temp;
  int vb;
  if (!pc)
    vb=0;
  else
    vb=1;
  while (px)
    { temp=new mat;
      if (!vb)
         { cplc=temp;
           vb=1;
         }
      temp->lin=px->lin;
      temp->col=px->col;
      temp->info=px->info;
      pc->urm=temp;
      pc=temp;
      px=px->urm;
    }
  pc->urm=NULL;
}
void main()
{ clrscr();
  printf("\nDati nr. de linii si coloane pt. matricea A si B: ");
  scanf("%d",&n1);
  pm1=new matrice[10]; pm2=new matrice[10];
  contor1=contor2=0;
  vb1=vb2=0;
  printf("\nDati elem. matricei A: \n");
  for(int i=0;i<n1;i++)
    for(int j=0;j<n1;j++)
        { printf(" A[%1d][%1d]=",i,j);
          scanf("%d",&pm1[i][j]);
          if (!(*pm1[i][j]))
             contor1++;
        }
  printf("\nDati elem. matricei B: \n");
  for(i=0;i<n1;i++)
  for(int j=0;j<n1;j++)
      { printf(" B[%1d][%1d]=",i,j);
        scanf("%d",&pm2[i][j]);
        if (!(*pm2[i][j]))
             contor2++; }
  printf("\nMatricea A este: \n");
  for(i=0;i<n1;i++)
    { for(int j=0;j<n1;j++)
           printf(" %d ",*pm1[i][j]);
      printf("\n");
    }
```

```
  printf("\nDin %d elemente, %d elemente sunt zero",n1*n1,contor1);
  getch();
  printf("\nMatricea B este: \n");
  for(i=0;i<n1;i++)
     { for(int j=0;j<n1;j++)
           printf(" %d ",*pm2[i][j]);
       printf("\n");
     }
  printf("\nDin %d elemente, %d elemente sunt zero",n1*n1,contor2);
  getch();
  cpla=creare(pm1);
  printf("\n\nMatricea A devine:\n");
  tipareste(cpla);
  getch();
  cplb=creare(pm2);
  printf("\n\nMatricea B devine:\n");
  tipareste(cplb);
  getch();
  delete pm1; delete pm2;
  pa=cpla; pb=cplb;
  vb1=0;
  cplc=pc=NULL;
  while ( (pa)&&(pb) )
      { if (pa->lin<pb->lin)
           { pc=new mat;
             pc->lin=pa->lin;
             pc->col=pa->col;
             pc->info=pa->info;
             pc->urm=NULL;
             if (!vb1)
               { cplc=pc;
                 vb1=1;
                 prar=pc;
                }
             else
               { prar->urm=pc;
                                    prar=pc;
                             };
                  pa=pa->urm;
                   }
            else
              if (pa->lin>pb->lin)
                  { pc=new mat;
                    pc->lin=pb->lin;
                    pc->col=pb->col;
                    pc->info=pb->info;
                    pc->urm=NULL;
                   if (!vb1)
                     { cplc=pc;
                       vb1=1;
                       prar=pc;
                      }
                   else
                     { prar->urm=pc;
                       prar=pc;
                     };
                   pb=pb->urm;
                  }
           else
             If (pa->col<pb->col)
                    { pc=new mat;
```

```
                        pc->lin=pa->lin;
                        pc->col=pa->col;
                        pc->info=pa->info;
                        pc->urm=NULL;
                        if (!vb1)
                          { cplc=pc;
                            vb1=1;
                            prar=pc;
                           }
                        else
                            { prar->urm=pc;
                              prar=pc;
                            };
                       pa=pa->urm;
                     }
              else
                if (pa->col>pb->col)
                      { pc=new mat;
                        pc->lin=pb->lin;
                        pc->col=pb->col;
                        pc->info=pb->info;
                        pc->urm=NULL;
                        if (!vb1)
                              { cplc=pc;
                                vb1=1;
                                prar=pc;
                              }
                        else
                             { prar->urm=pc;
                               prar=pc;
                             };
                       pb=pb->urm;
                       }
                  else
                    { pc=new mat;
                      pc->lin=pb->lin;
                               pc->col=pb->col;
                               pc->info=pb->info+pa->info;
                               pc->urm=NULL;
                              if (!vb1)
                                  { cplc=pc;
                                    vb1=1;
                                    prar=pc; }
                              else
                                 { prar->urm=pc;
                                   prar=pc;
                                 };
                              pa=pa->urm;
                              pb=pb->urm;
                         }
             }
  if (!pa)
      if (pb)
         copie(pb,pc);
  if (!pb)
      if (pa)
         copie(pa,pc);
  printf("\n\nMatricea C este: \n");
  tipareste(cplc);
  getch();
}
```

Problema LS6: să se construiască o listă liniară simplu înlănţuită şi apoi să se scrie funcţiile de inserare a unui element în listă, la sfârşitul listei şi în interiorul ei.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
         { int info;
           lista *urm;
         };
lista *cpl,*p,*p1;
void scriere(lista *pp)
{ printf("\n");
  while (pp)
      { printf(" %d ",pp->info);
        pp=pp->urm;
       } }
lista *inser1(lista *p1,int k)
{ lista *pp;
  pp=new lista;
  pp->info=k;
  pp->urm=p1;
  return pp;
}
lista *inser2(lista *p1,int k)
{ lista *pp,*p=p1;
  pp=new lista;
  pp->info=k;
  pp->urm=NULL;
  if (p)
     { while (p->urm)
           p=p->urm;
         p->urm=pp;
       }
   else
       p1=p;
  return p1;
  }
lista *inser3(lista *p1,int n,int k)
{ lista *pp=p1,*q;
  for(int i=0;i<n-1;i++)
     p1=p1->urm;
  q=new lista;
  q->info=k;
  q->urm=p1->urm;
  p1->urm=q;
  return pp;
}
void main()
{ clrscr();
  printf("\nIntroduceti elem. listei: \n");
  p=new lista;
  cpl=p;
  scanf("%d",&p->info);
  for(int i=0;i<4;i++)
     { p1=new lista;
       scanf("%d",&p1->info);
       p->urm=p1;
```

```
       p=p1;
     }
  p->urm=NULL;
  scriere(cpl);
  cpl=inser1(cpl,5);
  scriere(cpl);
  cpl=inser2(cpl,10);
  scriere(cpl);
  cpl=inser3(cpl,3,15);
  scriere(cpl);
  getch();
}
```

Problema LS7: să se construiască o listă dublu înlănţuită şi să se scrie elementele acesteia în ambele sensuri în funcţie de opţiune. Să se şteargă apoi un element al listei şi să se tipărească lista rezultată.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct listad
       { int x;
         listad *prec,*urm;
       };
void scrie(listad *pp,int i)
{ while (pp)
     { printf(" %d ",pp->x);
       if (!i)
         pp=pp->urm;
       else
         pp=pp->prec;
      }
}
void main()
{ listad *p1,*p2,*p3,*p4,*pw;
  clrscr();
  p1=new listad; p2=new listad;
  p3=new listad; p4=new listad;
  p1->x=3; p2->x=5;
  p3->x=11; p4->x=21;
  p1->prec=NULL; p1->urm=p2;
  p2->prec=p1; p2->urm=p3;
  p3->prec=p2; p3->urm=p4;
  p4->prec=p3; p4->urm=NULL;
  pw=p1;
  printf("\nInainte de stergere: \n");
  scrie(pw,0);
  pw=p4;
  scrie(pw,1);
  p2->urm=p4;
  p4->prec=p2;
  pw=p1;
  printf("\nDupa stergere: \n");
  scrie(pw,0);
  pw=p4;
  scrie(pw,1);
  getch();
}
```

Problema LS8: să se costruiască o listă dublu înlănţuită cu 5 elemente citite dintr-un vector. Să se scrie funcţia recursivă de tipărire a listei în ambele sensuri. Să se scrie funcţia de ştergere a unui element din listă.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct listad
    { listad *p_anterior;
      listad *p_urmator;
      int val1,val2;
    };
const int x[5]={1,10,100,1000,10000};
const int y[5]={20000,2000,200,20,2};
listad *pprec,*pcurent,*purm,*ppp_pr,*ppp_ur,*pelem;
void tipareste(listad *ppreced,listad *purmat)
{ if (ppreced)
    { printf(" %d ",ppreced->val1);
      tipareste(ppreced->p_urmator,NULL);
    };
  if (purmat)
    { printf(" %d ",purmat->val2);
      tipareste(NULL,purmat->p_anterior);
    };
 }
listad *cauta(listad *ppreced,int wval)
{ listad *pp;
  while ((ppreced->val1)!=wval)
  ppreced=ppreced->p_urmator;
  return (ppreced);
}
void sterge(listad *ppreced)
{ ppreced->p_anterior->p_urmator=ppreced->p_urmator;
  ppreced->p_urmator->p_anterior=ppreced->p_anterior;
  delete ppreced;
}
void main()
{ clrscr();
  pprec=new listad;
  ppp_pr=pprec;
  pprec->p_anterior=NULL;
  pprec->val1=x[0];
  pprec->val2=y[0];
  for(int i=1;i<5;i++)
     { pcurent=new listad;
       pprec->p_urmator=pcurent;
       pcurent->p_anterior=pprec;
       pcurent->val1=x[i];
       pcurent->val2=y[i];
       pprec=pcurent; }
  pcurent->p_urmator=NULL;
  ppp_ur=pcurent;
  printf("\nLista dublu inlantuita este: \n");
  tipareste(ppp_pr,ppp_ur);
  pelem=cauta(ppp_pr,100);
  sterge(pelem);
  printf("\nDupa stergerea unui element: \n");
  tipareste(ppp_pr,ppp_ur);
  getch();
}
```

Problema LS9: definiţi structura de date corespunzătoare stocării unei matrice de forma:

```
     0 0 0 0 0
     0 0 0 0 0              1 1 1
A=   0 0 0 0 0          B= 1 1 1
     0 0 0 0 0              1 1 1
     0 0 0 0 0

    2 2 2
C=                  D=2 2 2
    2 2 2
```

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct structura
      { int nrlin,nrcol;
        int val;
        structura *pstr;
      };
structura *cpl,*p1,*p2;
int linii;
int vb=0;
void main()
{ clrscr();
  printf("\nNr. linii: "); scanf("%d",&linii);
  while (linii)
     { if (!vb)
         { p1=new structura;
           cpl=p1;
           vb=1;
           p1->nrlin=linii;
           printf("Nr. coloane: "); scanf("%d",&p1->nrcol);
           printf("Valoare: "); scanf("%d",&p1->val);
         }
       else
         { p2=new structura;
           p2->nrlin=linii;
           printf("Nr. coloane: "); scanf("%d",&p2->nrcol);
           printf("Valoare: "); scanf("%d",&p2->val);
           p1->pstr=p2;
           p1=p2;
          }
        printf("\nNr. linii: "); scanf("%d",&linii);
      }
  p1->pstr=NULL;
  p1=cpl;
  while (p1)
      {printf("\nnr.linii %d nr.coloane %d valoare %d " ,p1->nrlin,p1-
>nrcol ,
        p1->val);
        p1=p1->pstr;
       }
  getch();
}
```

Problema LS10: scrieţi funcţia nerecursivă care efectuează copierea unei liste, cu eliminarea elementelor care au ca informaţie utilă o valoare mai mică decât un nivel specificat de parametru.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
    { int info;
      lista *urm;
     };
lista *cp1,*cp2,*p1,*p2;;
int nr,val;
lista *creare(int nr)
{ lista *p1;
  if (!nr)
     return NULL;
  else
     { nr--;
       p1=new lista;
       printf("Dati informatia: ");
       scanf("%d",&p1->info);
       p1->urm=creare(nr);
       return p1;
     }
  }
void tiparire(lista *pp)
{ printf("\n");
  while (pp)
      { printf(" %d ",pp->info);
        pp=pp->urm;
      }
   }
lista *copiere(lista *pp,int k)
{ lista *p1,*p2,*cp2;
  int vb=0;
  cp2=NULL;
  if (!pp)
    return NULL;
  else
    { while (pp)
        { if (pp->info>k)
            { p2=new lista;
              p2->info=pp->info;
              if (!vb)
                { cp2=p2;
                  vb=1;
                  p1=p2;
                }
              else
                { p1->urm=p2;
                  p1=p2; } }
         pp=pp->urm;
       }
     if (cp2)
         p1->urm=NULL;
     return cp2;
   }
}
void main()
```

```
{ clrscr();
  printf("\nDati nr. de elem. ale listei: ");
  scanf("%d",&nr);
  cp1=creare(nr);
  printf("\nLista initiala este: ");
  tiparire(cp1);
  printf("\nDati valoarea parametrului: ");
  scanf("%d",&val);
  cp2=copiere(cp1,val);
  printf("\nNoua lista este: ");
  tiparire(cp2);
  getch();
}
```

Problema LS11: modelaţi şi programaţi servirea "peste rând" la distribuirea unor produse deficitare, folosind o structură de date de tip listă liniară simplu înlănţuită.

Rezolvare:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
struct lista
      { char nume[20];
        lista *urm;
      };
lista *cp1=NULL;
int nr=0;
char opt;
lista *adaug(lista *pp)
{ lista *p1,*p2;
  p1=new lista;
  p1->urm=NULL;
  nr++;
  printf("\nPozitia %d Nume client: ",nr);
  fflush(stdin); gets(p1->nume);
  if (!pp)
     return p1;
  else
    { p2=pp;
      while (pp->urm)
          pp=pp->urm;
     pp->urm=p1;
     return p2;
    }
}
void tipar(lista *pp)
{ int n=1;
  while (pp)
    { printf("\n%d %s",n++,pp->nume);
      pp=pp->urm; } }
lista *servire(lista *pp)
{ lista *p1,*p2,*p;
  int poz;
  if (!nr)
     { printf("Nu sunt clienti in lista! ");
       return pp;
     }
  else
     { printf("\nServim clientul de pe pozitia: "); scanf("%d",&poz);
```

```
       if (poz>nr)
         { printf("\nUltimul client se afla pe poz.%d ",nr);
           return pp;
         }
       else
        if (poz==1)
           { printf("\nServim normal pe clientul.%s ",pp->nume);
             nr--;
             p1=pp->urm;
             delete pp;
             return p1;
           }
        else
           { p1=p2=pp; p=pp;
             for(int i=0;i<poz-1;i++)
                  { p2=p1;
                    p1=p1->urm;
                  }
           printf("Servim peste rand pe clientul. "); puts(p1->nume);
           p2->urm=p1->urm;
           delete p1;
           return p;
           }
      }
 }
void main()
{ clrscr();
  do
   { printf("\nOptiuni de lucru: ");
     printf("\n - A - adaugare client la lista ");
     printf("\n - S - servire client ");
     printf("\n - P - tiparire lista ");
     printf("\n - Q - terminare lucru \n");
     printf("Optiunea:"); fflush(stdin); scanf("%c",&opt);
     switch (opt=toupper(opt))
       { case'A': cp1=adaug(cp1); break;
         case'S': cp1=servire(cp1); break;
         case'P': tipar(cp1); break;
       }
    } while (opt=='A'||opt=='S'||opt=='P') ;
}
```

Problema LS12: numărați valorile pozitive, negative şi nule ale informaţiei utile dintr-o listă. Folosiţi o funcţie care după apelare returnează un pointer la un vector cu trei componente ce stochează rezultatele parcurgerii listei.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
      { int b;
        lista *urm;
      };
typedef int vec[3];
typedef vec *pv;
lista *cpl;
int n;
vec vect;
pv pvec;
```

```
lista *creare(int n)
{ lista *prad;
  if (!n)
    return NULL;
  else
   { n--;
    prad=new lista;
    printf("Dati elem.: "); scanf("%d",&prad->b);
    prad->urm=creare(n);
    return prad;
   }
}
void tiparire(lista *pp)
{ printf("\n");
  while (pp)
    { printf(" %d ",pp->b);
      pp=pp->urm;
    }
}
pv numara(lista *p1)
{ pv plucru;
  plucru=&vect;
  for(int i=0;i<3;i++)
    (*plucru[i])=0;
  while (p1)
    { if (p1->b>0) (*plucru[0])++;
      else if (p1->b<0) (*plucru[1])++;
           else (*plucru[2])++;
     p1=p1->urm; };
   return plucru;
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. ale listei: "); scanf("%d",&n);
  cpl=creare(n);
  pvec=numara(cpl);
  printf("\nNr de elem. pozitive ale listei este: %d ",*pvec[0]);
  printf("\nNr de elem. negative ale listei este: %d ",n-
(*pvec[0]+*pvec[2]));
  printf("\nNr de elem. nule ale listei este: %d ",*pvec[2]);
  getch();
 }
```

Problema LS13: scrieţi programul pentru inserarea unui element într-o listă simplu înlănţuită ale cărei elemente sunt deja sortate crescător, pe poziţia care păstrează lista ordonată.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
     { int info;
       lista *urm;
     };
lista *cpl,*pnou;
lista *pp,*pp1,*pp2;
char rasp;
int k,vb;
lista *creare(char r)
{ lista *p1;
```

```
  int k1;
  if (r=='n')
    return NULL;
 else
  { printf("\nDati un elem.: "); scanf("%d",&k1);
    p1=new lista;
    p1->info=k1;
    printf("\nIntroduceti elem.?[d/n] ");
    r=getche();
    p1->urm=creare(r);
    return p1;
  }
}
void afisare(lista *p)
{ printf("\n");
  if (!p)
   printf("\nLista vida");
  else
   while (p)
    { printf(" %d ",p->info);
      p=p->urm;
    }
 }
void main()
{ clrscr();
  printf("Construiti lista?[d/n] "); scanf("%c",&rasp);
  cpl=creare(rasp);
  afisare(cpl);
  printf("\nDati elem. de inserat: "); scanf("%d",&k);
  if (!cpl)
    printf("\nLista vida");
  else
    if (k<cpl->info)
       { pp=new lista;
         pp->urm=cpl;
         cpl=pp;
         pp->info=k;
         printf("\nElem. a fost inserat\n ");
       }
    else
      { vb=0;
        pp1=cpl;
        pp2=pp1->urm;
        while ( (pp2)&&(!vb) )
            { if ( (pp1->info<k)&&(pp2->info>k))
                 { pnou=new lista;
                   pnou->urm=pp2;
                   pp1->urm=pnou;
                   pnou->info=k;
                   vb=1;
                 }
              else if ( (pp1->info==k)||(pp2->info==k)) vb=2;
                   pp1=pp2;
                   pp2=pp2->urm;
             }
    }
  if (vb==1)
     printf("\nElem. a fost inserat\n ");
  else
     if (vb==2) printf("\nElem. exista deja ");
     else if (!pp2->urm)
```

```
           { pnou=new lista;
             pp1->urm=pnou;
             pnou->info=k;
             pnou->urm=NULL;
             printf("\nElem. a fost inserat\n ");
           }
     afisare(cpl);
  getch();
}
```

Problema LS14: scrieţi şi apelaţi funcţia care concatenează o listă cu o copie a sa.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
struct lista
       { int a;
         lista *urm;
       };
int nr;
lista *cap,*cap1,*cap2;
lista *creare(int nr)
{ lista *p;
  if (!nr)
     return NULL;
  else
    { nr--;
      p=new lista;
      printf("informatia: ");
      scanf("%d",&p->a);
      p->urm=creare(nr);
      return p;
    }
}
void tiparire(lista *p)
{ printf("\n");
  while (p)
    { printf(" %d ",p->a);
      p=p->urm;
    }
}
lista *concatenare(lista *p1,lista *p2)
{ lista *p=p1;
  if (!p1)
     return p2;
  else
    if (!p2)
      return p1;
    else
      { while (p->urm)
          p=p->urm;
        p->urm=p2;
        return p1; } }
lista *copiere(lista *pp1)
{ lista *p,*p1,*p2;
  if (!pp1)
    return NULL;
 else
```

```
  { p1=new lista;
    p1->a=pp1->a;
    p=p1;
    pp1=pp1->urm;
    while (pp1)
       { p2=new lista;
         p2->a=pp1->a;
         p1->urm=p2;
         p1=p2;
         pp1=pp1->urm;
       }
    p1->urm=NULL;
    return p; } }
void main()
{ clrscr();
  printf("\nNumar elem.: "); scanf("%d",&nr);
  cap1=creare(nr);
  printf("\nLista initiala este: ");
  tiparire(cap1);
  cap2=copiere(cap1);
  printf("\nLista copie: ");
  tiparire(cap2);
  cap=concatenare(cap1,cap2);
  printf("\nLista concatenata cu copia ei este: ");
  tiparire(cap);
  getch();
}
```

Problema LS15: scrieţi şi apelaţi funcţia care verifică dacă matricea rară pătratică ale cărei elemente sunt stocate într-o listă, conţine numai elemente ale diagonalei principale.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
struct lista
       { int lin;
         int col;
         int val;
         lista *urm;
       };
int nr;
lista *cpl;
lista *creare(int nr)
{ lista *pnou;
  if (!nr)
    return NULL;
  else
    { nr--;
      pnou=new lista;
      printf("Linia: "); scanf("%d",&pnou->lin);
      printf("Coloana: "); scanf("%d",&pnou->col);
      printf("Valoarea: "); scanf("%d",&pnou->val);
      pnou->urm=creare(nr);
      return pnou;
    }
}
verifica(lista *pl)
{ while (pl)
```

```
    if ( (pl->lin)!=(pl->col) )
        { return (1);
          pl=pl->urm; }
    else
       { return (0);
         pl=NULL;
       }
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. nenule: "); scanf("%d",&nr);
  printf("\nDati elem. nenule ale matricei: \n");
  cpl=creare(nr);
  if (!verifica(cpl))
    printf("\nMatricea contine numai elem. diagonalei principale ");
  else
    printf("\nMatricea nu contine numai elem. diagonalei principale
");
  getch();
}
```

Problema LS16: scrieţi şi apelaţi funcţia care returnează lungimea în baiţi a unei liste liniare simplu înlănţuite.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
struct lista
      { int info;
        lista *urm;
      };
int n,lg;
lista *cpl;
lista *creare(int n)
{ lista *ps1,*p1,*p2;
  p1=new lista;
  printf("Dati primul elem. al listei: ");
  scanf("%d",&p1->info);
  ps1=p1;
  for(int i=2;i<=n;i++)
       { p2=new lista;
         printf("Dati elem.: ");
         scanf("%d",&p2->info);
         p1->urm=p2;
         p1=p2; }
  p1->urm=NULL;
  return ps1;
}
int lungime(lista *pl)
{ int nr=0;
  while (pl)
     { nr++;
       pl=pl->urm;
     }
  return ((nr*sizeof(lista))+sizeof(cpl));
}
void tiparire(lista *pp)
{ printf("\n");
  while (pp)
```

```
    { printf(" %d ",pp->info);
      pp=pp->urm;
    }
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. al listei: "); scanf("%d",&n);
  cpl=creare(n);
  printf("\nElementele listei sunt: ");
  tiparire(cpl);
  lg=lungime(cpl);
  printf("\nLungimea in BYTE a listei este: %d ",lg);
  getch(); }
```

Problema LS17: scrieţi programul care realizează sortarea unei liste simplu înlănţuite cu conservarea vechilor legături.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
      { int info;
       lista *urm; };
lista *cp1,*p1;
int nr,k=1,temp;
lista *creare(int nr)
{ lista *p1;
  if (!nr)
    return NULL;
  else
    { nr--;
      p1=new lista;
      printf("Dati informatia: ");
     scanf("%d",&p1->info);
     p1->urm=creare(nr);
     return p1;
   }
}
void main()
{ clrscr();
  printf("\nDati nr. de elem. ale listei: ");
  scanf("%d",&nr);
  cp1=creare(nr);
  p1=cp1;
  while (k)
    { k=0;
      p1=cp1;
      while (p1)
         { if ((p1->info) > (p1->urm->info))
             { temp=p1->info;
               p1->info=p1->urm->info;
               p1->urm->info=temp;
               k++;
             }
          p1=p1->urm;
         }
     }
  p1=cp1;
  printf("\nSir sortat: ");
  while(p1)
```

```
      { p1=p1->urm;
        printf(" %d ",p1->info);
      }
  getch();
}
```

Problema LS18: să se scrie lista formată din elementele L(1), L(2),...L(n). Calculaţi cu o funcţie care returnează un vector, sumele: S(i)= $\sum_{j=1}^{i} L(j).util$, pe care le veţi afişa la apelare.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct lista
      { int util;
        lista *urm;
      };
typedef int s[20];
typedef s *ps;
lista *cp1;
int n,i;
s ss;
ps pss;
lista *constr(int n)
{ lista *ps1,*p1,*p2;
  p1=new lista;
  printf("Dati primul elem. al listei: ");
  scanf("%d",&p1->util);
  ps1=p1;
  for(int i=2;i<=n;i++)
      { p2=new lista;
        printf("Dati elem. %d al listei: ",i);
        scanf("%d",&p2->util);
        p1->urm=p2;
        p1=p2; }
  p1->urm=NULL;
  return ps1;}
void parcurgere(lista *p1)
{ while (p1)
     { printf(" %d ",p1->util);
       p1=p1->urm;
     } }
ps calcul(lista *p1)
{ int h=0;
  i=1;
  while (p1)
    { h+=p1->util;
      ss[i]=h;
      printf("\nSuma %d =%d",i++,h);
      p1=p1->urm;
    }
  pss=&ss;
  return pss;
}
void corp()
{ cp1=constr(n);
  parcurgere(cp1);
  calcul(cp1);}
```

```
void main()
{ clrscr();
  printf("\nDati n: ");
  scanf("%d",&n);
  corp();
  getch();}
```

Stive şi cozi

Problema SC1: se realizează evidenţa materialelor existente într-o magazie. La intrarea în stoc a materialelor pe bază de factură, se adaugă elemente unei stive. La eliberare spre consum productiv, se şterge vârful stivei. Procesul continuă după cum avem intrări sau consumuri de materiale. Se va scrie şi funcţia pentru numărarea facturilor.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>
typedef struct stiva{
                  int cant;
                  stiva * paep;
                    }STIVA;
unsigned char optiune;
int cantt,lung;
STIVA * vs=0;
STIVA *adaug(STIVA *p1, int cant){
  STIVA *p2;
  p2=(STIVA *)malloc(sizeof(STIVA));
  p2->cant=cant;
  p2->paep=p1;
  return p2;
}
STIVA *sterge(STIVA *p1){
  STIVA *p2;
  if(!p1) return 0;
  else{
      printf("\n%d",p1->cant);
      p2=p1->paep;
      free(p1);
      return p2; } }
int lungime(STIVA *p1){
  int n=0;
  while(p1){
            n++;
            p1=p1->paep;
           }
  return n; }
void tiparire(STIVA * p1){
  printf("\n");
  while(p1){
      printf(" %d ",p1->cant);
      p1=p1->paep;
           }
}
void main(){
  clrscr();
  printf("\nOptiune[a/s/n/p/t]: ");
```

```
  scanf("%c",&optiune);
  do{
    switch((tolower(optiune))){
       case 'a': printf("\nCantitate: ");
                 scanf("%d",&cantt);
                 vs=adaug(vs,cantt);
                 break;
       case 's': vs=sterge(vs);
                 break;
       case 'n': lung=lungime(vs);
                 printf("\nLungime: %d",lung);
                 break;
       case 'p':tiparire(vs);
                 break;
       case 't':return;
      default: printf("\nParametru necunoscut!");
     }
  printf("\nOptiune [a/s/n/p/t]: ");fflush(stdin);
  scanf("%c", &optiune);
  }while(optiune!='t');
}
```

Problema SC2: se consideră o stivă nevidă de lungime >5. Scrieţi formula pentru dezalocarea memoriei ocupată de această stivă.

Rezolvare:

```
#include<conio.h>
#include<stdio.h>
#include<alloc.h>
typedef struct stiva{
                    int cod;
                    int cant;
                    int pret;
                    stiva *paep;
                    }STIVA;
STIVA *vs=0;
int lung,cantt,codd,prett,n;
STIVA * adaug(STIVA *p1,int codd,int cantt, int prett){
  STIVA * p2;
  p2=(STIVA *)malloc(sizeof(STIVA));
  p2->cod=codd;
  p2->cant=cantt;
  p2->pret=prett;
  p2->paep=p1;
  return p2;
}
STIVA *sterge(STIVA* p1){
  STIVA *p2;
  if(!p1) return 0;
  else{
   printf("\n%d",p1->cant);
   p2=p1->paep;
   free(p1);
   return p2;
      }
}
int lungime(STIVA* p1){
  int n=0;
  while(p1){
             n++;
```

```
            p1=p1->paep;
          }
  return n;
}
void main(){
  printf("\nNr de elemente in stiva ");
  scanf("%d",&n);
  while(n<5){
     printf("\nNr elemente in stiva ");
     fflush(stdin);scanf("%d",&n);
          }
  for(int i=1;i<=n;i++){
      printf("\ncod: ");
      scanf("%d",&codd);
      printf("\ncantitate: ");
      scanf("%d",&cantt);
      printf("\npret: ");
       scanf("%d",&prett);
      vs=adaug(vs,codd,cantt,prett);
      }
  printf("\nlungime initiala= %d",lungime(vs));
  for(i=1;i<=n;i++) vs=sterge(vs);
  printf("\nlungime dupa stergere= %d",lungime(vs));
}
```

Problema SC3: o echipă realizează demontarea unei instalaţii cu dispunerea reperelor şi subansamblurilor într-un mod corespunzător reparării şi mai apoi asamblării. O altă echipă efectuează asamblarea. Scrieţi programul care afişează lista operaţiilor de asamblare, ştiind că aceste operaţii sunt specificate prin numele reperelor sau subansamblelor.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct stiva{
                     char nume[30];
                     stiva *paep;
                    }STIVA;
STIVA *p1,*p2;
unsigned char raspuns;
void main(){
  p1=(STIVA *)malloc(sizeof(STIVA));
  p1->paep=0;
  raspuns='d';
  while((raspuns=='d')||(raspuns=='D')){
     printf("\nNume reper: ");
     scanf("%s",p1->nume);
     p2=(STIVA *)malloc(sizeof(STIVA));
     p2->paep=p1;
     p1=p2;
     printf("\nContinuati[d/n]? ");fflush(stdin);
     scanf("%c",&raspuns);
    }
  p2=p1->paep;
  free(p1);
  while(p2){
     printf("\nmontare %s",p2->nume);
     p2=p2->paep;
     }
```

```
}
```

Problema SC4: se consideră mulţimea literelor mari şi mici. Să se construiască pornind de la un şir oarecare de litere două stive ce conţin literele mari şi mici. Să se afişeze literele mici şi mari în ordinea în care au fost introduse.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct stiva{
                    unsigned char litere;
                    stiva * paue;
                                        }STIVA;
STIVA *p1,*p2,*p11,*p22,*p4,*p5;
unsigned char ch;
STIVA *copiere_stiva(STIVA *p){
  STIVA *p3=0,*p31;
  while(p){
     p31=(STIVA *)malloc(sizeof(STIVA));
     p31->litere=p->litere;
     p31->paue=p3;
     p3=p31;
     p=p->paue;
   }
  return p3;
}
void sterge_stiva(STIVA *p){
  STIVA *pp=p;
  while(pp){
    p=pp->paue;
    free(pp);
    pp=p;
  }
}
void listare_stiva(STIVA *p){
  while(p){
     printf("\n%c",p->litere);
     p=p->paue;
 } }
void main(){
  p1=p2=0;
  printf("\nIntroduceti cate un caracter/sau*: ");fflush(stdin);
  scanf("%c",&ch);
  while(ch!='*'){
       if((ch<='z')&&(ch>='a')){
          p11=(STIVA*)malloc(sizeof(STIVA));
          p11->litere=ch;
          p11->paue=p1;
          p1=p11;
          }
       if((ch<='Z')&&(ch>='A')){
          p22=(STIVA*)malloc(sizeof(STIVA));
          p22->litere=ch;
          p22->paue=p2;
          p2=p22;
              }
       scanf("%c",&ch);
   }
```

```
  p4=copiere_stiva(p1);
  p5=copiere_stiva(p2);
  sterge_stiva(p1);
  sterge_stiva(p2);
  printf("\nStiva cu literele mici: ");
  listare_stiva(p4);
  printf("\nStiva cu literele mari: ");
  listare_stiva(p5);
}
```

Problema SC5: scrieţi şi apelaţi funcţia de ştergere a unei stive până la elementul a cărui informaţie utilă corespunde cu o informaţie dată.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>
typedef struct stiva{
                      int info;
                      stiva *prec;
                     }STIVA;
STIVA *vs=0;
int param;
char* c;
STIVA *cr_stiva(STIVA *pp){
  STIVA *p;
  p=(STIVA *)malloc(sizeof(STIVA));
  printf("\nInformatia utila: ");
  scanf("%d",&p->info);
  p->prec=pp;
  return p;
}
STIVA *sterge(STIVA *ps,int param){
  STIVA *p;
  while((ps->info!=param)&&(ps)){
          p=ps->prec;
          free(ps);
          ps=p;
   }
  return ps;
}
void afisare(STIVA *ps){
  while(ps){
      printf("\n%d",ps->info);
      ps=ps->prec;
   }
}
void main(){
  do{
     vs=cr_stiva(vs);
     printf("\nApasati <ENTER> pt.introducerea urm.element ");
     fflush(stdin);
     scanf("%c",c);
  }while(isspace(*c));
  afisare(vs);
  printf("\nIntroduceti parametrul: ");
  scanf("%d",&param);
  vs=sterge(vs,param);
  afisare(vs);
```

```
  getch();
}
```

Problema SC6: dându-se stivele A şi B care conţin cod produs, cantitate şi respectiv cod produs şi preţ, creaţi stiva ale cărei elemente sunt presupuse sortate după cod.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct stivaa{
                      int cod_prod;
                    int cant;
                    stivaa *paep;
                    }STIVAA;
STIVAA *pstivaa;
typedef struct stivab{
                     int cod_prod;
                     int pret;
                     stivab *paep;
                     }STIVAB;
STIVAB *pstivab;
typedef struct stivac{
                      int cod_prod;
                      int cant;
                      int pret;
                      stivac *paep;
                      }STIVAC;
STIVAC *pstivac;
STIVAA *p1,*vs1;
STIVAB *p2,*vs2;
STIVAC *p3,*pp,*vs;
int n;
STIVAA *creare1(STIVAA *ps1){
  p1=(STIVAA *)malloc(sizeof(STIVAA));
  printf("\ncod produs: ");
  scanf("%d",&p1->cod_prod);
  printf("\ncantitate: ");
  scanf("%d",&p1->cant);
  p1->paep=ps1;
  return p1;
}
STIVAB *creare2(STIVAB *ps2){
  p2=(STIVAB *)malloc(sizeof(STIVAB));
  printf("\npret: ");
  scanf("%d",&p2->pret);
  p2->paep=ps2;
  return p2;
}
void tiparire(STIVAC *ps){
  puts("\n");
  while(ps){
     printf(" %d ",ps->cod_prod);
     printf(" %d ",ps->cant);
     printf(" %d ",ps->pret);
     printf("\n");
     ps=ps->paep;
     }
 }
```

```
void main(){
  printf("\nDati n: ");
  scanf("%d",&n);
  vs1=0;vs2=0;
  for(int i=1;i<=n;i++){
      vs1=creare1(vs1);
      vs2=creare2(vs2);
     };
  p1=vs1;p2=vs2;
  pp=0;
  while(p1&&p2){
      vs=(STIVAC *)malloc(sizeof(STIVAC));
      vs->cod_prod=p1->cod_prod;
      vs->cant=p1->cant;
           vs->pret=p2->pret;
           vs->paep=pp;
           p1=p1->paep;
           p2=p2->paep;
           pp=vs;
          }
   printf("\n");
   tiparire(vs);
  }
```

Problema SC7: descompuneţi folosind o funcţie pe care o apelaţi, o stivă în două stive cu număr egal de elemente, numai dacă acest lucru este posibil.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct stiva{
                    int util;
                    stiva *adr;
                    }STIVA;
STIVA *vsm1,*vsm=0,*pvm;
int n;
int numara(STIVA * v){
  STIVA *vas=v;
  int as;
  if(!v) return 0;
  else
    if(vas) as=(1+(numara(vas->adr)));
    else return as; }
STIVA *divizare(STIVA *v1,int nn){
  STIVA *divih;
  for(int k=nn;k>1;k--){
                        v1=v1->adr;
                       }
  divih=v1->adr;
  v1->adr=0;
  return divih;
}
void tiparire(STIVA *v3){
  if(!v3)
    printf("\nstiva vida!");
  else{
    while(v3){
         printf(" %d ",v3->util);
```

```
          v3=v3->adr;
               }
    printf("\n");
      }
}
void main(){
  clrscr();
  STIVA *va;
  printf("\nDati nod stiva(pt sfarsit tastati '0'): ");
  scanf("%d",&n);
  while(n){
      pvm=(STIVA *)malloc(sizeof(STIVA));
      pvm->util=n;
       pvm->adr=vsm;
       vsm=pvm;
       printf("\nDati nod stiva(pt sfarsit tastati '0'): ");
       scanf("%d",&n);
      }
  int nr=numara(vsm);
  printf("\n");
  printf("\nNr de elemente din lista initiala: %d",nr);
  printf("\n");
  if(!(nr%2)){
        nr=nr/2;
        vsm1=divizare(vsm,nr);
        printf("\nSTIVA 1: ");
        tiparire(pvm);
        printf("\nSTIVA 2: ");
        tiparire(vsm1);
             }
  else printf("\nStiva nu contine nr par de elemente! ");
  printf("\n");
  getch();
}
```

Arbori binari şi arbori de căutare

Problema AB1: să se scrie programul care realizează:

- creare arbore binar de căutare;
- parcurgere arbore;
- inserare nod în arbore;
- ştergere nod din arbore;
- determinarea înălţimii arborelui;
- determinarea greutăţii arborelui;
- verificarea dacă este arbore binar complet sau nu;

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct arb
          { int info;
            arb*st,*dr;
          };
arb *root=NULL,*p;
int nr,nr1,nr2,i;
char c;
void inserare(arb *&r,int k)
```

```
{ arb *pp;
  if (!r)
     { pp=new arb;
       pp->info=k;
       pp->st=NULL;
       pp->dr=NULL;
             r=pp;
            }
  else
     if (r->info>k)
            inserare(r->st,k);
     else
         if (r->info<k)
              inserare(r->dr,k);
         else
              printf("Elementul deja exista! \n");
}
void sterg( arb *&r)
{ if (r->dr)
    sterg(r->dr);
  else
    { arb *a=r;
      r=r->st;
     delete a;
   }
}
void stergere( int x, arb *&p)
{ arb *q;
  if (!p)
     printf("\nElement negasit! \n");
  else
     if (x<p->info)
        stergere(x,p->st);
     else
        if (x>p->info)
           stergere(x,p->dr);
        else
            { q=p;
              if (!q->dr)
                    { p=q->st;
                      delete q;
                    }
               else
                 if (!q->st)
                      { p=q->dr;
                        delete q;
                      }
                 else
                   sterg(q->st); }
}
int max(int x,int y)
{ if (x<y ) return y;
   else return x;
}
int nivel(arb *r)
{ if (!r)
    return 0;
  else
    return (1+max( nivel(r->st),nivel(r->dr) ));
}
void preordine(arb *r)
```

```
{ if (r)
    { printf(" %d ",r->info);
      preordine(r->st);
      preordine(r->dr);
    }
}
void inordine(arb *pp)
{ if (pp)
    { inordine(pp->st);
      printf(" %d ",pp->info);
      inordine(pp->dr);
    }
}
void postordine(arb *pp)
{ if (pp)
   { postordine(pp->st);
     postordine(pp->dr);
     printf(" %d ",pp->info);
   }
}
int frunza(arb *pp)
{ return ( (!pp->st)&&(!pp->dr) );
}
int nrfrunze(arb *r)
{ if (!r) return 0;
  else
     if (frunza(r))
               return 1;
     else
        return (nrfrunze(r->st)+nrfrunze(r->dr) );
}
int putere(int nr,int k)
{ if (!k)
     return 1;
  else
     return (nr*putere(nr,k-1));
}
int numara(arb *r)
{ if (!r)
    return 0;
else
    return ( 1 + numara(r->st) + numara(r->dr) ); }
void main()
{ clrscr();
  printf("Nr.:"); scanf("%d",&nr);
  while (nr!=-1)
     { inserare(root,nr);
       printf("Nr.:");
       scanf("%d",&nr); }
  inordine(root); printf("\n");
  postordine(root); printf("\n");
  preordine(root); printf("\n");
  printf("\nCe nod doriti sa stergeti? ");
  scanf("%d",&nr);
  stergere(nr,root);
  preordine(root); printf("\n");
  printf("\nVERIFICARE ARBORE\n ");
  nr=nrfrunze(root);
  nr1=numara(root);
  printf("\nArborele are %d noduri ",nr1);
  nr1=numara(root);
```

```
  printf("\nGreutatea arborelui (noduri terminale): %d ",nr);
  i=nivel(root);
  printf("\nInaltimea arborelui: %d ",i);
  nr2=putere(2,i);
  if (nr1==(nr2-1))
      printf("\nArbore complet - contine nr. max. de noduri pt.
inaltimea sa ");
  else
    printf("\nArborele nu este complet ");
  getch();
}
```

Problema AB2: să se scrie funcţia care realizează copierea unui arbore şi apoi să se afişeze nodurile acestuia.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct arb
          { int val;
            arb *st;
            arb *dr;
          };
arb *rad,*stg,*drt,*pstg,*pdrt;
int val;
arb *constr_nod(arb *st,arb *dr,int val)
{ arb *pnod;
  pnod=new arb;
  pnod->st=st;
  pnod->dr=dr;
  pnod->val=val;
  return pnod;
}
void tiparire(arb *r)
{ if (r)
    { printf("\n %d ",r->val);
      tiparire(r->st);
      tiparire(r->dr);
    }
}
arb *copie(arb *r)
{ if (!r)
      return NULL;
  else return constr_nod(r->st,r->dr,r->val); }
void main()
{ clrscr();
  stg=constr_nod(NULL,NULL,111);
  drt=constr_nod(NULL,NULL,112);
  pstg=constr_nod(stg,drt,11);
  stg=constr_nod(NULL,NULL,121);
  drt=constr_nod(NULL,NULL,122);
  pdrt=constr_nod(stg,drt,12);
  rad=constr_nod(pstg,pdrt,1);
  tiparire(rad);
  stg=copie(rad);
  printf("\nNoul arbore este:");
  tiparire(stg);
  getch();
}
```

Problema AB3: să se construiască un arbore binar cu elemente citite dintr-un vector, să se scrie o funcţie care returnează adresa unui anumit nod al arborelui, nod de care se va lega un alt arbore construit cu elemente citite tot dintr-un vector.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct arb
         { int val;
           arb *st;
           arb *dr;
         };
typedef int mat[7];
mat y={111,112,11,121,122,12,1} ;
mat z={200,300,400,500,600,700,800} ;
arb *rad,*stg,*drt,*pstg,*pdrt,*p=NULL;
int val;
arb *constr_nod(arb *st,arb *dr,int val)
{ arb *pnod;
  pnod=new arb;
  pnod->st=st;
  pnod->dr=dr;
  pnod->val=val;
  return pnod;
}
arb *constr( mat x)
{ stg=constr_nod(NULL,NULL,x[0]);
  drt=constr_nod(NULL,NULL,x[1]);
  pstg=constr_nod(stg,drt,x[2]);
  stg=constr_nod(NULL,NULL,x[3]);
  drt=constr_nod(NULL,NULL,x[4]);
  pdrt=constr_nod(stg,drt,x[5]);
  return constr_nod(pstg,pdrt,x[6]);
}
void cautare(arb *pnod,int pval)
{ arb *nod;
  if (pnod)
    if (pnod->val==pval)
         p=pnod;
    else
       { cautare(pnod->st,pval);
         cautare(pnod->dr,pval);
       }
}
void tiparire(arb *r)
{ if (r)
      { printf(" %d ",r->val);
        tiparire(r->st);
        tiparire(r->dr);
      }
}
arb *copie(arb *r)
{ if (!r)
     return NULL;
  else
     return constr_nod(r->st,r->dr,r->val);
}
void main()
{ clrscr();
```

```
  rad=constr(y);
  printf("\nradacina=%d",rad->val);
  cautare(rad,111);
  if (p)
     { printf("\nvaloare=%d",p->val);
        p->dr=constr(z);
        }
  printf("\n");
  tiparire(rad);
  getch();
}
```

Problema AB4: să se construiască un arbore binar, să se scrie două funcţii care calculează suma elementelor din subarborele stâng respectiv suma elementelor din subarborele drept.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct arb
         { int util1,util2;
           arb *st,*dr;
         };
arb *pp;
arb *constr_nod(arb *st,int u1,int u2,arb *dr)
{ arb *p;
  p=new arb;
  p->util1=u1;
  p->util2=u2;
  p->st=st;
  p->dr=dr;
  return p; }
arb *constr()
{ arb *f1,*f2,*nod;
  f1=constr_nod(NULL,1,2,NULL);
  f2=constr_nod(NULL,3,4,NULL);
  nod=constr_nod(f1,5,6,f2);
  f1=constr_nod(NULL,7,8,NULL);
  f2=constr_nod(nod,9,10,f1);
  f1=constr_nod(NULL,11,12,NULL);
  nod=constr_nod(f1,13,14,f2);
  return nod;
}
int suma1(arb *rad)
{ if (!rad)
        return 0;
  else
        return ( suma1(rad->st) + rad->util1 + suma1(rad->dr) );
}
int suma2(arb *rad)
{ if (!rad)
     return 0;
  else
      return ( suma2(rad->st) + rad->util2 + suma2(rad->dr) );
}
void main()
{ clrscr();
  pp=constr();
  printf("\n %d ",suma1(pp));
  printf("\n %d ",suma2(pp));
```

```
  getch();
}
```

Problema AB5: să se construiască un arbore binar în care informaţia utilă este alcătuită dintr-o valoare întreagă şi un pointer la o listă simplu înlănţuită. Să se scrie funcţia de construire listă, construire arbore, tipărire listă, tipărire arbore şi de căutare a unui nod din arbore cu o valoare dată.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<alloc.h>
struct lista
          { char nume[30];
            lista *urm;
           };
struct arb
          { int val;
            lista *capl;
            arb *st,*dr;
          };
arb *pp,*pcaut=NULL;
int cod;
void listare(lista *cap)
{ while(cap)
    { printf(" %s ",cap->nume);
      cap=cap->urm;
    }
  printf("\n");
}
lista *inssr(lista *cap,char st[20])
{ if (cap)
    { cap->urm=inssr(cap->urm,st);
      return cap;
    }
  else
    { lista *p=(lista*)malloc(sizeof(lista));
      strcpy(p->nume,st);
      p->urm=NULL;
      return p;
    };
}
lista *constr_lista()
{ lista *cp=NULL;
  char str[20];
  while (printf("Dati nume:"),fflush(stdin),gets(str),strcmp(str,"*"))
      cp=inssr(cp,str);
  return cp;
}
arb *constr_nod(arb *st,int val1,lista *cap,arb *dr)
{ arb *p;
  p=new arb;
  p->val=val1;
  p->capl=cap;
  p->st=st;
  p->dr=dr;
  return p;
}
lista *concatenare(lista *cap1,lista *cap2)
```

```
{ lista *pp;
  if (!cap1)
       return cap2;
   else
      { pp=cap1;
        while (cap1->urm)
        cap1=cap1->urm;
        cap1->urm=cap2;
        return pp;
      }
}
arb *constr_arb()
{ arb *b1,*b2,*c1,*c2,*b,*c,*nod;
  lista *capb1,*capb2,*capc1,*capc2,*capa,*capb,*capc;
  printf("\n\nConstruiti lista b1: \n");
  capb1=constr_lista();
  listare(capb1);
  printf("\n\nConstruiti lista b2:\n ");
  capb2=constr_lista();
  listare(capb2);
  printf("\n\nConstruiti lista c1:\n ");
  capc1=constr_lista();
  listare(capc1);
  printf("\n\nConstruiti lista c2:\n ");
  capc2=constr_lista();
  listare(capc2);
  b1=constr_nod(NULL,21,capb1,NULL);
  b2=constr_nod(NULL,22,capb2,NULL);
  c1=constr_nod(NULL,31,capc1,NULL);
  c2=constr_nod(NULL,32,capc2,NULL);
  capb=concatenare(capb1,capb2);
  printf("\n\nlista capb:\n");
  listare(capb);
  capc=concatenare(capc1,capc2);
  printf("\n\nlista capc:\n");
  listare(capc);
  b=constr_nod(b1,2,capb,b2);
  c=constr_nod(c1,3,capc,c2);
  capa=concatenare(capb,capc);
  printf("\n\nlista capa:\n");
  listare(capa);
  nod=constr_nod(b,1,capa,c);
  return nod;
}
void caut_nod(arb *rad,int elem)
{ if (rad)
  if (rad->val==elem)
        pcaut=rad;
  else
       { caut_nod(rad->st,elem);
         caut_nod(rad->dr,elem);
       }
}
void tiparire(arb *r)
{ if (r)
     { printf(" %d ",r->val);
       listare(r->capl);
       tiparire(r->st);
       tiparire(r->dr);
     }
}
```

```
void main()
{ clrscr();
  pp=constr_arb();
  printf("\n\nArborele este:\n "); tiparire(pp);
  printf("\ndati cod: "); scanf("%d" ,&cod);
  caut_nod(pp,cod);
  if (pcaut)
     listare(pcaut->capl);
  else
     printf("\nElement negasit in arbore! ");
  getch();
}
```

Problema AB6: scrieţi şi apelaţi funcţia de copiere a unui arbore binar.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct nod
          { int val;
            nod *st,*dr;
          };
nod *rad1=NULL,*rad2,*rr;
int nr;
void creare(nod *&r,int k)
{ nod *pp;
  if (!r)
      { pp=new nod;
        pp->val=k;
        pp->st=NULL;
        pp->dr=NULL;
        r=pp;
      }
  else
     if (r->val>k)
         creare(r->st,k);
     else
         if (r->val<k)
            creare(r->dr,k);
         else
             printf("Elementul deja exista! \n");
}
void parcurg(nod *rad)
{ if (rad)
     { parcurg(rad->st);
       printf(" %d ",rad->val);
       parcurg(rad->dr);
     }
}
nod *copiere(nod *pp)
{ nod *q;
  if (pp)
      { q=new nod;
        q->val=pp->val;
        q->st=copiere(pp->st);
        q->dr=copiere(pp->dr);
        return q;
      }
  else
```

```
      return NULL;
}
void main()
{ clrscr();
  while( printf("Nr.:"),( scanf("%d",&nr)!=EOF) )
      creare(rad1,nr);
  printf("\nArborele 1 este: \n");
  parcurg(rad1);
  rr=copiere(rad1);
  printf("\nArborele 2 este: \n");
  parcurg(rr);
  getch();
}
```

Fişiere

Problema FS1: să se scrie programul carerealizează afişarea pe ecran a conţinutul unui fişier de tip text. Numele fişierului se dă în linia de comandă. Pentru execuţia programului am folosit un fişier de testare FISIER.TST. Lansarea în execuţie a programului se face C:\>Prob81 FISIER.TST

Rezolvare:

```
#include<stdio.h>
#include<stdlib.h>
#include<CONIO.H>
void main(int argc,char *paramstr[]){
  FILE*pf;
  int c;
  clrscr();
  if(argc==2){
        if(!(pf=fopen(paramstr[1],"r")))
            {printf("\nNu se poate deschide fisierul!");
             exit(1);}
        puts("\n");
             }
 else {printf("\nNr necorespunzator de argumente in linia de
comanda!");
  exit(1);}
  while((c=getc(pf))!=EOF) putc(c,stdout);
  fclose(pf);
}
```

Problema FS2: să se scrie programul care codifică şi decodifică conţinutul unui fişier. Programul va citi numele fişierului din linia de comandă şi în funcţie de o opţiune:

- codifică conţinutul fişierului specificat şi noul fişier rezultat va avea atributul Hidden, adică va fi fişier ascuns (nu va putea fi vizualizat cu comanda DIR);
- decodifică un fişier criptat şi îi setează atributul de normal;

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<io.h>// pt prototipul functiei unlink
#include<string.h>
```

```
#include<process.H>
void main(int argc,char *paramstr[]){
  clrscr();
  unsigned char opt;
  int c;
  FILE *fp,*fp1;
  if(argc==2){
      if(!(fp=fopen(paramstr[1],"r"))){
              printf("\nNu se poate deschide fisierul!");
              exit(1);}
      puts("\n");
      if(!(fp1=fopen("TEMP.TXT","w")))
            {printf("\nNu se poate deschide fisierul!");
             exit(1);}
           }
  else{
     printf("nr necorespunzator de parametri in linia de comanda!");
     exit(1);}
  printf("\nCriptare/Decriptare (c/d) ?");
  fflush(stdin);
  scanf("%d",&opt);
  while((c=getc(fp))!=EOF){
        if(c>125)putc((char(c-125)),fp1);
        if(c<125)putc((char(c+125)),fp1);
        }
  printf("\ns-a criptat/decriptat");
  fclose(fp);
  fclose(fp1);
  unlink(paramstr[1]);
  rename("TEMP.TXT",paramstr[1]);
  if(opt=='c')
     _dos_setfileattr(paramstr[1],FA_HIDDEN);
  else _dos_setfileattr(paramstr[1],_A_HIDDEN );
  getch();
}
```

Problema FS3: să se scrie un program care determină frecvenţa de apariţie a caracterelor ASCII dintr-un fişier, numele fişierului citindu-se din linia de comandă.

Rezolvare:

Se va folosi o structură de tip stivă. Algoritmul de calcul este următorul:

- se citeşte caracter cu caracter din fişierul specificat până se ajunge la sfârşitul fişierului;
- se verifică dacă acest caracter există deja în stivă:
  - dacă nu există acest caracter se pune în stivă iniţializând numărul de apariţii cu 1;
  - dacă există se incrementează numărul de spaţii;
- când s-a ajuns la sfârşit de fişier, se ordonează stiva în funcţie de numărul de apariţii a fiecărui caracter;

Rezolvare:

```
#include <stdio.h>
#include<string.h>
#include<conio.h>
#include <alloc.h>//malloc
#include<stdlib.h>//malloc
typedef struct tnod{
```

```
                  unsigned char info;
                  int nr;
                  struct tnod*paep;
                   }TNOD;
typedef enum{false,true} boolean;
boolean vb;
int contor1=0;
int contor2=0;
TNOD * vs;
FILE * fp;
unsigned char c;
TNOD * inserare( TNOD* ps, unsigned char cc){
  TNOD * p;
  p=(TNOD*)malloc(sizeof(TNOD));
  p->info=cc;
  p->nr=1;
  p->paep=ps;
  return(p);
}
boolean verificare(TNOD *ps, unsigned char cc){
  if(ps==NULL)return false;
  else{
     while((ps->info!=cc)&&(ps->paep!=0))ps=ps->paep;
  if(ps->info==cc){
        (ps->nr)++;
         return true;
         }
  else return false;
    }
}
void scriere(TNOD* ps){
  if(!ps)
        printf("\nStiva vida");
  else while(ps){
          switch(ps->info){
                         case 13:printf("\n CR %d",ps->nr);
                         case 10:printf("\n LF %d",ps->nr);
                        default:printf("\n%c %d",ps->info,ps->nr);
                         }
          ps=ps->paep;
           }
}
void numarare(TNOD * ps){
  while(ps){
           contor1+=ps->nr;
           contor2++;
           ps=ps->paep;
          }
}
TNOD * ordonare(TNOD *ps){
  unsigned char vb;
  TNOD *p1,*p2,*temp;
  vb=0;
  p1=ps;
  while(p1){
       p2=p1->paep;
       while(p2!=0){
          if(p1->nr<p2->nr){
                temp->info=p1->info;
                temp->nr=p1->nr;
                p1->info=p2->info;
```

```
                p1->nr=p2->nr;
                p2->info=temp->info;
                p2->nr=temp->nr;
                              }
            p2=p2->paep;
                 }
      if(!vb){
         ps=p1;
         vb=1;
            }
      p1=p1->paep;
      }
  return ps;
}
void main(int argc,char *paramstr[]){
  clrscr();
  FILE*fp;
  int c;
  vs=0;
  if(argc==2){
     if((fp=fopen(paramstr[1],"r"))==0)
        {printf("\nNu se poate deschide fisierul!");
         exit(1);}
    puts("\n");
              }
  else {printf("\nNr necorespunzator de argumente in linia de
comanda!");
  exit(1);}
  while((c=getc(fp))!=EOF){
      vb=verificare(vs,c);
      if(!vb) vs=inserare(vs,c);
                           }
  fclose(fp);
  scriere(vs);
  vs=ordonare(vs);
  printf("\nStiva ordonata!");
  scriere(vs);
  numarare(vs);
  printf("\nNr de caractere din fisier: %d",contor1);
printf("\nNr de caractere distincte: %d",contor2);
}
```

Obiecte şi clase

Problema OC1: să se definească un obiect ce conţine o dată de tip articol şi o procedură. Articolul conţine la rândul lui un vector de 10 elemente de tip întreg. Să se afişeze elementele vectorului folosind un pointer la articol şi un pointer la obiect. Să se apeleze procedura definită în cadrul obiectului.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
Typedef int a[10];
struct b { a c; };
class d { public:
          struct b e;
          void beta();
```

```
        };
typedef b *pb;
typedef d *pd;
void d:: beta()
   { printf("\n aaa "); } ;
a x;
pb pp;
Pd ppp;
void main()
{ clrscr();
  for (int i=0;i<10; i++)
     x[i]=(i+1)*(i+1)*(i+1);
  pp=(b*)x;
  ppp=(d*)x;
  for(i=0;i<10;i++)
      printf("\n%5d %5d %5d",x[i],(*pp).c[i],(*ppp).e.c[i]);
  (*ppp).beta();
  getch();
}
```

Problema OC2: care este diferenţa între cele două programe de mai jos? Ce rezultate afişează fiecare dintre ele?

Programul 1:

```
#include<stdio.h>
#include<conio.h>
class a { public:
          int x;
          int y;
          int suma (int u,int v);
        };
class b { public:
          a w;
          int z;
          int difer(a s,int t);
        };
int a::suma(int u,int v)
{ return u+v; };
int b::difer(a s,int t)
{ return (w.suma(s.x,s.y)-t*t); };
a gg;
b jj;
int aa,bb;
void main()
{ clrscr();
  gg.x=100; gg.y=222;
  jj.w.x=11; jj.w.y=33; jj.z=20;
  aa=gg.suma(gg.x,gg.y);
  bb=jj.difer(gg,jj.z);
  printf("\n aa=%d",aa);
  printf("\n bb=%d",bb);
  getch();
}
```

Programul 2:

```
#include<stdio.h>
#include<conio.h>
class a { public:
          int x;
          int y;
```

```
         int suma (int u,int v);
        };
class b { public:
          a w;
          int z;
          int difer(a s,int t);
         };
int a::suma(int u,int v)
{ return u+v; };
int b::difer(a s,int t)
{ return (w.suma(s.x,s.y)-t*t); };
a gg;
b jj;
int aa,bb;
void main()
{ clrscr();
  gg.x=100; gg.y=222;
  jj.w.x=11; jj.w.y=33; jj.z=20;
  aa=gg.suma(gg.x,gg.y);
  bb=jj.difer(jj.w,jj.z);
  printf("\n aa=%d",aa);
  printf("\n bb=%d",bb);
  getch();
}
```

Problema OC3: indicaţi ce afişează programul:

```
#include<stdio.h>
#include<conio.h>
typedef int a[10];
struct b
       { a c;};
class f { public:
          a p;
          void alfa();
         };
void f:: alfa()
{ printf(" progr. orientata pe obiecte "); };
a x,*y;
b m,n;
f v,w;
void main()
{ clrscr();
  for (int i=0;i<10; i++)
     { x[i]=(i+1)*(i+1);
        m.c[i]=(i+1)*(i+1);
       v.p[i]=(i+1)*(i+1);
      };
  y=&x;
  for(i=0;i<10;i++)
     printf("y[%d]=%4d\n",i,(*y)[i]);
  printf("\n");
  n=m; w=v;
  for(i=0;i<10;i++)
     printf("n.c[%d]=%4d w.p[%d]=%4d\n",i,n.c[i],i,w.p[i]);
  w.alfa();
  getch();
}
```

Rezolvare:
În urma execuţiei se va afişa:

Y[0] = 1
Y[1] = 4
Y[2] = 9
Y[3] = 16
Y[4] = 25
Y[5] = 36
Y[6] = 49
Y[7] = 64
Y[8] = 81
Y[9] = 100
n.c[0] = 1 w.p[0] = 1
n.c[1] = 4 w.p[1] = 4
n.c[2] = 9 w.p[0] = 9
n.c[3] = 16 w.p[0] = 16
n.c[4] = 25 w.p[0] = 25
n.c[5] = 36 w.p[0] = 36
n.c[6] = 49 w.p[0] = 49
n.c[7] = 64 w.p[0] = 64
n.c[8] = 81 w.p[0] = 81
n.c[9] = 100 w.p[0] = 100
progr. orientata pe obiecte

Problema OC4: să se definească un vector de obiecte pentru calcularea mediilor studenţilor dintr-o grupă.
Rezolvare:

```
#include<stdio.h>
#include<conio.h>
typedef int vec[5];
struct student
         { char nume[20];
           int varsta;
           vec note;
           float med;
         };
class studobj { public:
              struct student datt;
              float medie(vec xx);
             void scrie(struct student yyy); };
float studobj:: medie(vec xx)
{ float xm=0;
  for(int i=0;i<5;i++)
        xm+=xx[i];
  xm/=5;
  return xm;
} ;
void studobj::scrie( struct student yyy)
{ printf("\n %s media:%5.2lf ",yyy.nume,yyy.med); };
studobj stt[10];
int nn;
float xx,med;
void main()
{ clrscr();
```

```
  printf("\nnumar studenti [1..10]: ");
  scanf("%d",&nn);
  for (int ii=0;ii<nn; ii++)
        { printf("nume:");
          fflush(stdin); gets(stt[ii].datt.nume);
          printf("varsta:");
          scanf("%d",stt[ii].datt.varsta);
          for(int jj=0;jj<5;jj++)
              { printf("nota - %d :",jj+1);
               fflush(stdin); scanf("%d",&stt[ii].datt.note[jj]);
              }
         stt[ii].datt.med=stt[ii].medie(stt[ii].datt.note);
        };
  for(ii=0;ii<nn;ii++)
       stt[ii].scrie(stt[ii].datt);
  getch();
}
```

Problema OC5: definiţi un obiect numit muncitor, având o structură (ce descrie numele, salariul orar, numărul de ore şi reţinerile), având o variabilă de tip întreg pentru descrierea salariului şi două proceduri şi o funcţie (pentru citire date, scriere rezultat şi calcul salariu). Scrieţi programul care într-o structură repetitivă apeleză cele două proceduri şi funcţia pentru calculul salariului pentru un număr de muncitori.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct pers
       { char nume[20];
         int nrore;
         int salorar;
         int retineri;
       };
class muncitor { public:
              struct pers persoana;
              int salariu;
              void citeste(pers &ppersoana);
              int calcul(pers wpersoana);
              void scrie(pers tpersoana,int tsalariu);
               };
typedef muncitor *pmuncitor;
void muncitor::citeste(pers &ppersoana)
{ printf("\nnume:"); fflush(stdin); gets(ppersoana.nume);
  printf("nr. ore lucrate:"); scanf("%d",&ppersoana.nrore);
  printf("salariu orar:"); scanf("%d",&ppersoana.salorar);
  printf("retineri:"); scanf("%d",&ppersoana.retineri);
};
Int muncitor::calcul(pers wpersoana)
{ return (wpersoana.nrore*wpersoana.salorar-wpersoana.retineri); };
void muncitor::scrie(pers tpersoana,int tsalariu)
{ printf("\n%s salariu: %d\n",tpersoana.nume,tsalariu); };
Int nr,sal;
pmuncitor pmuncit;
void main()
{ clrscr();
  printf("\nnumar de muncitori: ");
  scanf("%d",&nr);
  pmuncit=new muncitor;
  for (int i=0;i<nr; i++)
```

```
     { pmuncit->citeste(pmuncit->persoana);
       sal=pmuncit->calcul(pmuncit->persoana);
       pmuncit->scrie(pmuncit->persoana,sal);
     };
  delete pmuncit;
  getch();
}
```

Problema OC6: definiţi obiectul stoc material şi calculaţi valoarea totală a materialelor fără mişcare, conţinute de un vector de obiecte, iniţializate de la terminal.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
struct mat
        { int cod;
          int cant;
          int pret;
          int intr;
          int ies;
        };
class stocmat { public:
            struct mat aaa;
            int valstoc(int c1,int p1);
              };
int stocmat:: valstoc(int c1,int p1)
{ return (c1*p1); };
stocmat vec[100];
int a,n;
int valtot=0;
void main()
{ clrscr();
  printf("\ndati numarul de materiale: ");
  scanf("%d",&n);
  for (int i=0;i<n; i++)
       { printf("\ndati cod material: "); scanf("%d",&vec[i].aaa.cod);
         printf("dati cantitatea:"); scanf("%d",&vec[i].aaa.cant);
         printf("dati pretul: "); scanf("%d",&vec[i].aaa.pret);
         printf("dati intrarile:"); scanf("%d",&vec[i].aaa.intr);
         printf("dati iesirile: "); scanf("%d",&vec[i].aaa.ies);
         if ((vec[i].aaa.intr==0)&&(vec[i].aaa.ies==0))
             { a=vec[i].valstoc(vec[i].aaa.cant,vec[i].aaa.pret);
                          valtot+=a;
                        }
            };
 printf("valoare totala a materialelor fara miscare este: %d",valtot);
 getch();
}
```

Problema OC7: definiţi şi iniţializaţi o matrice. Referiţi aceasta structură printr-un pointer membru al unui obiect, într-o funcţie membru al aceluiaşi obiect.

Rezolvare:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef int mat[100][100];
```

```
typedef mat *pmat;
class nod { public:
           pmat c;
           void init(int n,int m);
           void list(int n,int m);
           int suma(int n,int m);
          };
void nod::init(int n,int m)
{ printf("\ndati elementele matricei: \n");
  c=new mat[1];
  for(int i=0;i<n;i++)
     for(int j=0;j<m;j++)
         scanf("%d",c[i][j]);
 };
void nod::list(int n,int m)
{ printf("\nelementele matricei sunt: ");
  for(int i=0;i<n;i++)
    for(int j=0;j<m;j++)
        printf("\n x(%d,%d)=%d ",i,j,*c[i][j]);
};
int nod::suma(int n,int m)
{ int s=0;
  for(int i=0;i<n;i++)
     for(int j=0;j<m;j++)
         s+=*c[i][j];
  return s;
};
nod p;
int n,m;
void main()
{ clrscr(); fflush(stdin);
  printf("dati nr. maxim de linii N[1..10]: "); scanf("%d",&n);
  printf("dati nr. maxim de coloane M[1..10]: "); scanf("%d",&m);
  p.init(n,m);
  printf("am initializat matricea ");
  p.list(n,m);
  printf("\nsuma elementelor matricei este: %d ",p.suma(n,m));
  getch();
}
```

Agregarea structurilor de date

Problema AG1: să se studieze problema migraţiei între judeţe a persoanelor, cu un nivel de detaliere corespunzător. Se consideră populaţia de sex masculin şi de sex feminin din mediul urban, respectiv, din mediul rural aparţinând celor 40 de judeţe şi Municipiului Bucureşti. Să se definească structura de date care permite calculul totalului subcolectivităţilor pe sexe şi pe medii din ţară cu utilizarea secvenţelor repetitive.

Rezolvare:

Se defineşte o matrice având 41 de linii şi 41 de coloane ce permite evidenţierea migraţiei populaţiei de la un judeţ la altul. Pentru defalcare pe sexe şi medii, componentele matricei vor fi la rândul lor câte o matrice cu două linii şi două coloane.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int migr[41][41][2][2];
  Clrscr();
  for(int i=0;i<41;i++)
     {
       for(int j=0;j<41;j++)
         {
           if(i!=j)
           {
        printf("De la judetul au migrat %d la judetul %d:\n",i+1,j+1);
        printf("In mediul urban %d persoane, din care %d M si %d
F\n",migr[i][j][0][0]+migr[i][j][0][1],migr[i][j][0][0],migr[i][j][0][
1]);
       printf("In mediul rural %d persoane, din care %d M si %d
F\n",migr[i][j][1][0]+migr[i][j][1][1],migr[i][j][1][0],migr[i][j][1][
1]);
           }
         }
     }
  getch();
}
```

Problema AG2: se consideră m colectivităţi ale căror elemente sunt descrise cu n caracteristici. Fiecare activitate are un număr oarecare de componente. Să se definească structura de date ce permite reprezentarea colectivităţilor în cazul particular în care n=2 (nivelul şi frecvenţa). Să se calculeze media şi dispersia pentru fiecare serie.

Rezolvare:

```
#include <stdio.h>
#include <conio.h>
struct Serie
               {
           int nc;
           int *val;
           int *frecv;
           };
float medie(Serie ser)
{
  float med=0;
  for(int i=0,s=0;i<ser.nc;i++)
     {
     med+=ser.val[i]*ser.frecv[i];
     s+=ser.frecv[i];
     }
  if(s!=0)
     med/=s;
  return med;
}
float dispersie(Serie ser,float (*pf)(Serie))
{
  float disp=0;
  float med=(*pf)(ser);
  for(int i=0,s=0;i<ser.nc;i++)
    {
     disp+=(ser.val[i]-med)*(ser.val[i]-med)*ser.frecv[i];
     s+=ser.frecv[i];
```

```
    }
  if(s!=0)
    disp/=s;
  return disp;
}
void main()
{
  Serie *ser;
  int m,n;
  clrscr();
  printf("\nNumarul de serii:");
  scanf("%d",&m);
  ser = new Serie[m];
  for(int i=0;i<m;i++)
    {
     printf("Numarul de caracteristici ale seriei %d: ",i+1);
     scanf("%d",&n);
     ser[i].nc=n;
     ser->val=new int[n];
     ser->frecv=new int[n];
     for(int j=0;j<n;j++)
      {
         printf("Caracteristica %d: ",j+1);
         scanf("%d",&ser[i].val[j]);
         printf("Frecventa %d: ",j+1);
         scanf("%d",&ser[i].frecv[j]);
       }
    }
  for(i=0;i<m;i++)
  {
   printf("\nMedia seriei %d este: %.3f",i+1,medie(ser[i]));
   printf("\nDispersia seriei %d este:
%.3f",i+1,dispersie(ser[i],medie));
  }
  for(i=0;i<m;i++)
  {
    delete [] ser[i].val;
     delete [] ser[i].frecv;
   }
  delete [] ser;
  getch();
}
```

Problema AG3: se consideră o mulţime formată din absolvenţii unei facultăţi, fiecare membru al colectivităţii având descrise notele la disciplinele din anii 1, 2, 3, 4 şi 5 de studii. Să se calculeze mediile anuale şi media generală a fiecărui absolvent. Datele se iniţializează la definire.

Rezolvare:

```
#include <stdio.h>
#include <conio.h>
#include <mem.h>
#define ANI 5
#define DISC 15
struct Note
            {
            int nr_discipline;
            float note[DISC];
            };
struct Student
```

```
            {
            int matricola;
            char nume[30];
            Note note_disc[ANI];
            };
void main()
{
  Student grup[]
  ={
  100,"Ion Ion",
  {{5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}}},
  101,"Petre Popescu",
  {{6,{8,8,8,8,8,10}},
  {6,{8,8,8,8,8,10}},
  {5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}},
  {5,{8,8,8,8.8,8}}},
  102,"Ion Mihai",
  {{5,{6,6,6,6,6}},
  {5,{7,7,7,7,7}},
  {5,{8,8,8,8,8}},
  {5,{9,9,9,9,9}},
  {5,{10,10,10,10,10}}}
  };
  float medii[ANI]={0,0,0,0,0};
  float med_gen=0.0;
  int n=sizeof(grup)/sizeof(Student);
  clrscr();
  printf("Matricola|Nume prenume|Medii ani| Media generala\n\n");
  for(int i=0;i<n;i++)
   {
    memset(medii,0,sizeof(float)*ANI);
       med_gen=0.0;
        for(int j=0;j<ANI;j++)
           {
              for(int k=0;k<grup[i].note_disc[j].nr_discipline;k++)
                 {
                  medii[j]+=grup[i].note_disc[j].note[k];
                 }
              medii[j]/=grup[i].note_disc[j].nr_discipline;
              med_gen+=medii[j];
           }
        med_gen/=ANI;
        printf("%d|%s|",grup[i].matricola,grup[i].nume);
        for(int k=0;k<ANI;k++) printf("%.2f|",medii[k]);
        printf("%.2f\n",med_gen);
  }
  getch();
}
```

Problema AG4: se consideră mulţimea facturilor emise într-o lună a cărei număr nu depăşeşte o valoare N, cunoscută. În fiecare factură se înscriu cel mult M produse sau servicii care se achiziţionează, indicându-se denumire, unitate de măsură,cantitate, preţ unitar, TVA. Să se scrie programul care calculează valoarea totală a produselor şi serviciilor achiziţionate într-o lună folosind vectori de articole, fiecărei facturi

corespunzându-i o componentă a unui vector şi fiecărui produs sau serviciu corespunzându-i, de asemenea, o componentă în alt vector de articole.

Rezolvare:

```
#include <stdio.h>
#include <conio.h>
#define N 30
#define M 20
struct ArticolFactura
                      {
                      char den[20];
                      int pu;
                      char um[4];
                      int cant;
                      int ctva;
                      };
struct Factura
              {
              int nra;
              int numar;
              char data[11];
              ArticolFactura af[M];
               };
void main()
{
  int n,m;
  Factura fact[N];
  double valf=0.0;
  double tvaf=0.0;
  double total_fara_tva=0.0;
  double total_tva=0.0;
  double total=0.0;
  clrscr();
  printf("Numar de facturi:");
  scanf("%d",&n);
  for(int i=0;i<n;i++)
     {
      valf=0.0;
      tvaf=0.0;
      printf("\nNumar factura: ");
      scanf("%d",&fact[i].numar);
      fflush(stdin);
      printf("Data:");
      gets(fact[i].data);
      printf("Articole in factura %d: ",i+1);
      scanf("%d",&fact[i].nra);
      fflush(stdin);
      for(int j=0;j<fact[i].nra;j++)
         {
          printf("\n\nDenumire:");
          gets(fact[i].af[j].den);
          printf("Unitatea de masura:");
          gets(fact[i].af[j].um);
          printf("Cantitatea:");
          scanf("%d",&fact[i].af[j].cant);
          printf("Pret unitar:");
          scanf("%d",&fact[i].af[j].pu);
          printf("Cota TVA:");
          scanf("%d",&fact[i].af[j].ctva);
          fflush(stdin);
```

```
          valf+=(double)fact[i].af[j].cant*fact[i].af[j].pu;
tvaf+=(fact[i].af[j].cant*fact[i].af[j].pu*fact[i].af[j].ctva)/(double
)100;
         }
     total_fara_tva+=valf;
     total_tva+=tvaf;
    }
   total=total_fara_tva+total_tva;
   printf("Total fara tva: %lf\nTotal tva: %lf\nValoare totala
facturi:
 %lf\n",total_fara_tva,total_tva,total);
  getch();
}
```

Problema AG5: se consideră firme dintr-un oraş ale căror tranzacţii sunt reflectate prin facturi emise. Să se calculeze valoarea totală a tranzacţiilor la nivelul fiecărei firme.

Rezolvare:

Întrucât nu este cunoscut numărul firmelor din oraş se va construi o listă care se completează cu noi elemente pe măsură ce se autorizează noi firme. Fiecare element al listei va conţine şi o variabilă pointer cu rol de cap de listă pentru a adăuga elemente pe măsură ce se emit facturi.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define NMAX 20
struct Factura
              {
              double val;
           Factura *urm_fact;
           };
struct Firma
          {
          char nume[30];
          Factura *prima_fact;
          Firma *urm;
          };
void main()
{
  Firma *f1,*f2,*f3;
  Factura *fa11,*fa12,*fa21,*fa22,*fa23,*fa31;
  Firma *tf=NULL;
  Factura *tfa=NULL;
  double sum=0.0;
  fa11=new Factura;
  fa11->val=3000000.0;
  fa12=new Factura;
  fa12->val=4000000.0;fa12->urm_fact=NULL;
  fa21=new Factura;
  fa21->val=1000000.0;
  fa22=new Factura;
  fa22->val=2000000.0;
  fa23=new Factura;
  fa23->val=3000000.0;fa23->urm_fact=NULL;
  fa31=new Factura;
  fa31->val=8000000.0;fa31->urm_fact=NULL;
  fa11->urm_fact=fa12;
  fa21->urm_fact=fa22;
```

```
  fa22->urm_fact=fa23;
  f1=new Firma;strcpy(f1->nume,"Firma 1");f1->prima_fact=fa11;
  f2=new Firma;strcpy(f2->nume,"Firma 2");f2->prima_fact=fa21;
  f3=new Firma;strcpy(f3->nume,"Firma 3");f3->prima_fact=fa31;f3-
  >urm=NULL;
  f1->urm=f2;
  f2->urm=f3;
  tf=f1;
  while(tf!=NULL)
    {
    tfa=tf->prima_fact;
    puts(tf->nume);
    sum=0.0;
    while(tfa!=NULL)
       {
       sum+=tfa->val;
       tfa=tfa->urm_fact;
       }
    printf("%.lf LEI\n",sum);
    tf=tf->urm;
    }
  delete f1;
  delete f2;
  delete f3;
  delete fa11;
  delete fa12;
  delete fa21;
  delete fa22;
  delete fa23;
  delete fa31;
  getch();
}
```

Problema AG6: se consideră o structură de date referitoare la produse având ca informaţii: denumire produs, cheltuielile fixe, cheltuieli variabile, precum şi o metodă de calcul a preţului produsului. Să se creeze o nouă structură de date care are în plus un adaos la preţul de bază obţinut şi o nouă funcţie de calcul a preţului produsului.

Rezolvare:

Se vor utiliza clase şi obiecte, evidenţiindu-se derivarea.

```
#include <iostream.h>
#include <conio.h>
class ProdusB
           {
   protected:
      float ChF, ChV;
     char Denumire[30];
public:
  ProdusB()
    {
     cout<<"Denumire produs:";
     cin>>Denumire;
    cout<<"Cheltuieli fixe:";
    cin>>ChF;
    cout<<"Cheltuieli variabile:";
    cin>>ChV;
}
float virtual CalculPret()
{
```

```
return ChF+ChV;
}
friend void AfisPret(ProdusB *p)
{
Cout<<"Pretul produsului "<<p->Denumire<<" este "
<<p->CalculPret()<<endl;
}
};
class ProdusD:public ProdusB
{
 private:
   float Adaos;
 public:
   ProdusD()
  {
   cout<<"Adaos:";
   cin>>Adaos;
  };
  float CalculPret()
  {
  float pg=ProdusB::CalculPret();
  return pg+Adaos/100*pg;
  };
};
void main()
{
  clrscr();
  ProdusB *pretr = new ProdusB();
  ProdusD *preta = new ProdusD();
  AfisPret(pretr);
  AfisPret(preta);
  getch();
}
```

Problema AG7: să se creeze o clasă ce descrie datele despre o persoană: nume, prenume şi adresă, adresa fiind la rândul ei o clasă, având ca date membre strada, numărul şi judeţul.

Rezolvare:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
class Adresa
             {
  Private:
     Char str[30];
     int nr;
     Char judet[30];
  public:
     Void SetStrada(const char *s)
      {
        strcpy(str,s);
      };
     Void SettNr(const int n)
       {
       Nr=n;
       }
    void SetJudet(const char *j)
     {
     strcpy(judet,j);
```

```
     }
    friend ostream & operator<<(ostream &out,const Adresa &a);
   Friend istream & operator>>(istream &in,Adresa &a);
   };
  Ostream & operator<<(ostream &out,const Adresa &a)
  {
   out<<"Adresa:"<<endl;
   out<<"Strada:"<<a.str<<" nr. "<<a.nr<<endl;
   out<<"Judetul:"<<a.judet<<endl;
  return out;
  }
  Istream & operator>>(istream &in,Adresa &a)
   {
   cout<<"Strada:";
   in>>a.str;
   cout<<"Nr: ";
   in>>a.nr;
   cout<<"Judetul:";
   in>>a.judet;
   Return in;
  }
class Persoana
       {
 private:
   char nume[30];
   char prenume[30];
   Adresa adr;
 public:
   Persoana();
   void AfiseazaDate()
  {
    cout<<"Nume:"<<nume<<endl;
    cout<<"Prenume:"<<prenume<<endl;
    cout<<adr;
  }
 };
Persoana::Persoana()
{
 cout<<"Nume:";
 cin>>nume;
 cout<<"Prenume:";
 cin>>prenume;
 cin>>adr;
}
void main()
{
  Persoana p1;
  p1.AfiseazaDate();
}
```