7. ARTICOLUL – STRUCTURĂ DE DATE NEOMOGENĂ ŞI CONTIGUĂ

7.1 Structuri de date poziţionale

Există o diversitate de date cu care se caracterizează indivizii unei colectivităţi. După efectuarea unei analize detaliate, se conchide că un număr *q* de caracteristici sunt suficiente pentru descrierea elementelor colectivităţii. În continuare numim şablon de descriere o anumită ordine în care sunt înşirate caracteristicile, ordine absolut necesară a fi respectată la descrierea fiecărui element al colectivităţii.

Fie caracteristicile C_1, C_2, ..., C_q dispuse în ordinea care se constituie în şablonul asociat descrierii indivizilor colectivităţii considerate.

De exemplu, pentru descrierea materialelor existente în stoc, se consideră caracteristicile:

C_1 – numele materialului;
C_2 – unitatea de măsură;
C_3 – cantitatea existentă în stoc la începutul perioadei;
C_4 – preţul unitar;
C_5 – data ultimei aprovizionări;
C_6 – intrările de materiale;
C_7 – ieşirile de materiale;
C_8 – codul materialului;
C_9 – stocul final.

deci $q = 8$. Şablonul pentru introducerea datelor are elementele:

$$C_8\ C_1\ C_2\ C_4\ C_3\ C_5\ C_6\ C_7 \qquad (7.1)$$

Caracteristica C_9 nu e necesară pentru că rezultă din calcule.

Aceste caracteristici sunt poziţionale, în sensul că C_8 este prima caracteristică în şablon, C_3 este a 5-a caracteristică în şablon, iar C_7 este ultima caracteristică a şablonului.

La introducerea datelor, şirurile de constante se constituie în câmpuri. Astfel, se identifică câmpul pentru codul materialului, format dintr-un număr fix de cifre, câmpul numele materialului, ce are un număr de caractere care nu depăşesc o valoare prestabilită etc.

Şablonul se descrie prin tabelul 7.1, unde lungimea este calculată ca număr de caractere.

Tabelul nr. 7.1 *Şablon de descriere a câmpurilor*

Caracteristică	Natură dată	Lungimea
C_8 – cod material	număr	5
C_1 – nume material	şir caractere	20
C_2 – unitate măsură	şir caractere	3
C_3 – cantitate existentă în stoc	număr	6
C_4 – preţ unitar	număr real	4+2
C_5 – data ultimei aprovizionări	şir caractere	9
C_6 – intrări	număr	6
C_7 – ieşiri	număr	6
C_9 – stoc final	număr	9

Se observă că cele opt caracteristici diferă ca natură, ca lungime a şirului de simboluri ce compun câmpurile şi ocupă în cadrul şablonului poziţii distincte.

Datele grupate în şablon cu poziţia fiecăreia specificată, formează o structură de date de tip articol.

Modelul grafic asociat articolului este prezentat în figura 7.1.

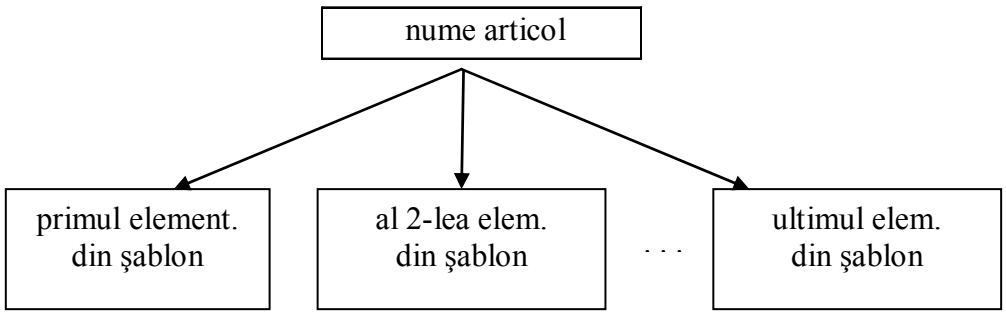

Figura 7.1 *Model grafic asociat şablonului de articol*

Se observă că pe ultimul nivel se află date elementare specificate prin nume şi tip. Structura, în ansamblul ei, are un nume şi se defineşte distinct, ceea ce urmează reprezentând componentele de la nivelul al doilea.

```
typedef struct material
{
          int cod;
          char nume[20];
          char um[3];
          float preț;
          int cont;
          int intrări;
          int ieşiri;
 };
```

$$lg(material) = lg(cod) + lg(nume) + lg(um) + + lg(pret) + lg(cont) + lg(intrari) + lg(iesiri) + \theta \qquad (7.2)$$

În cazul în care compilatorul face o alocare neoptimală θ reprezintă numărul biţilor impuşi de alinierea cerută de fiecare câmp precedent care are un alt tip decât întreg sau caracter. În cazul în care are loc optimizare în faza de compilare, θ devine nul.

$$\text{adr}(\text{câmp}_i) = \text{adr}(\text{câmp}_1) + \sum_{j=1}^{n-1} \text{lg}(\text{câmp j}) + \theta \qquad (7.3)$$

unde:

$$\theta_j = \begin{cases} 1 \; reprezint\check{a} \; baitul \; cerut \; de \; alinierea \; câmpului \; j+1 \\ 0 \; în \; cazul \; în \; care \; câmpj+1 \; nu \; cere \; alinierea \end{cases} \qquad (7.4)$$

Dacă datele de la nivelul al doilea le regăsim sub numele de câmpuri elementare, la nivelul superior data este numită dată de grup.

În unele situaţii fiecărui nivel *i* se asociază un număr, numit număr de nivel. Pentru nivelele inferioare se asociază numere naturale mai mari decât cel asociat nivelelor superioare. Unui nivel *i* se ataşează un număr sau un interval de numere. Descrierea în orice situaţie rămâne poziţională.

Astfel, pentru exemplul dat se folosesc descrieri cu numere de nivel:

01 material
02 cod
02 nume
02 um
02 preţ
02 cantitate
02 intrări
02 ieşiri
01 .....

Terminarea descrierii structurii, este marcată de apariţia unui număr de nivel care delimitează începutul altei structuri, sau o instrucţiune de program, proprie limbajului, care vine să marcheze începutul unei alte secvenţe program.

O altă descriere este:

7 material
8 cod
9 nume
10 um
8 preţ
10 cantitate
9 intrări
9 ieşiri
4 .....

unde, pentru primul nivel se utilizează numerele 4, 5, 6 sau 7, iar pentru al doilea nivel, sunt utilizate numerele 8, 9 sau 10.

La acest nivel de descriere a articolelor putem observa că o dată elementară este un articol cu câmpuri de acelaşi tip.

Astfel:

```
typedef struct a
{
     int b;
};

typedef struct c
{
       int c1;
       int c2;
       int c3;
```

```
        int c4;
        int c5;
};
```

şi

```
a x;
c w,y;
```

conduc la aceleaşi rezervări de zone de memorie pentru *x* şi *y* ca definiţiile:

```
int u;
int t[5],v[5];
```

$$adr(c4) = adr(c1) + lg(c1) + lg(c3) = adr(c1) + (4-1) * lg(int) \qquad (7.5)$$

iar

$$adr(v[4]) = adr(v[1]) + (4-1) * lg(int) \qquad (7.6)$$

Câmpurile unui articol se numesc membrii articolului, iar referirea lor se face folosind operatorul de referire „.”.

În exemplul considerat, *w* şi *y* sunt variabile de tip *c*, iar *t* şi *v* sunt masive, fiecare având 5 elemente. Prin $w.C_4$ se referă câmpul C_4 al variabilei *w*, iar prin $y.C_1$ se referă câmpul C_1 al variabilei *y*.

Dacă se ia în considerare tipul de variabilă articol *material* şi se face definirea:

```
material m;
```

m.cod, m.preţ sunt referiri ale câmpurilor ce alcătuiesc variabila *m* definită ca având tipul *material*.

Se observă că articolul este un tip de dată generic, pe care prin forma de definire a câmpurilor îl putem folosi pentru a obţine tipuri derivate de date.

Astfel, *material* şi *C* sunt tipuri derivate de date corespunzătoare tipului generic *struct*.

Definirea acestui tip pune în evidenţă:

- componentele, câmpurile care-l alcătuiesc;
- natura câmpurilor;
- poziţia câmpurilor.

Altfel spus, se evidenţiază structura sau şablonul datelor.

Variabilele definite pentru un tip derivat ocupă atâta memorie cât rezultă din lungimile câmpurilor însumate şi corectate cu alinierea şi au în componenţă exact aceleaşi elemente folosite la descrierea tipului, ca membri ai fiecărei variabile.

Descrierea unui tip derivat nu înseamnă rezervare de memorie, ci stocare de informaţie privind elementele care îl definesc.

Definirea variabilelor de tipul derivat descris anterior, efectuează rezervare de memorie.

7.2 Structuri de date ierarhizate

Datele de tip articol sunt ierarhizate pe două sau mai multe nivele. Pentru a obţine descrierea mai multor nivele este necesar ca la fiecare nivel inferior să fie posibilă enumerarea de date elementare sau date de grup. Datele de grup e necesar să fie descrise deja înainte de a le folosi, dar în unele cazuri particulare descrierea lor se efectuează ulterior folosirii.

Astfel, se definesc tipurile:

```
struct data
{
      int zi;
      int luna;
      int an;
};

struct adresa
{
     data data_stabilirii;
     char strada[30];
     int numar;
     char telefon[8];
     char oras[30];
};


struct  persoana
{
   data data_nasterii ;
   adresa locul_nasterii;
   int varsta;
   data data_angajarii;
   adresa adresa_actuala;
 };
```

Definirea ulterioară specificării datelor de grup, se face ca în exemplul:

01 persoana
02 data_stabilirii
03 zi
03 luna
03 an
02 locul_nasterii
03 data_nasterii
04 zi
04 luna
04 an
03 strada
03 numar

03 telefon
03 oras
02 varsta
02 data_angajarii
03 zi
03 luna
03 an
02 adresa_actuala
03 data_stabilirii
04 zi
04 luna
04 an
03 strada
03 numar
03 telefon
03 oras

Oricare ar fi modalitatea de descriere a structurii arborescente, aceasta trebuie să rămână în continuare neambiguă.

Dacă se acceptă ca definiţie a lungimii elementului de la nivelul *i*:

$$\lg(x,i)=\sum_{j=0}^{ni}(y,j,i+1) \qquad (7.7)$$

unde prin x_i înţelegem elementul y_j de pe nivelul *i+1*, astfel încât:

$$cont(x)\ ,\ adr(y_1) \qquad (7.8)$$

rezultă că adresa unui element y_j de pe nivelul *i* aparţinând datei de grup *k* dintr-o structură de tip articol, se obţine prin relaţia:

$$adr(y_j,0)=adr(y_1,i)+\sum_{k=1}^{k-1}(\lg(y_k,i)+\theta_k)+\sum_{h=1}^{j-1}(\lg(y_h,i)+\theta_k) \qquad (7.9)$$

De exemplu, pentru articolul definit prin:

```
struct student
{
      char nume[21];
      data data_nasterii;
      data data_admiterii;
};
```

adresa câmpului *data_admiterii.an* se obţine urmărind modelul grafic din figura 7.2

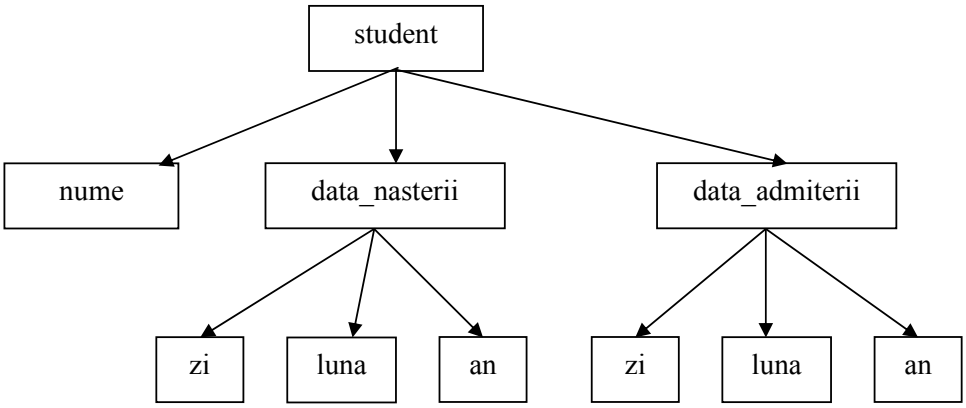

Figura 7.2 *Modelul grafic al structurii articolului student*

şi după formula:

$$
\begin{aligned}
adr(student.data\_admiterii.an,3) &= adr(student.nume,2)+ \\
&+ lg(student.nume,2)+\theta 1+ \\
&+ lg(student.data\_nasterii,2)+ \theta 2+ \\
&+ lg(student.data\_nasterii.zi)+ \theta 3+ \\
&+ lg(student.data\_nasterii.luna)+ \theta 4+ \\
&+ lg(student.nume,2)+ 21+\theta 1+4+4+4+ \\
&+\theta 2+4+\theta 3+4+\theta 4=41+1=42 \qquad (7.10)
\end{aligned}
$$

pentru că $\theta_2 = \theta_3 = \theta_4 = 0$.

Dacă se consideră punctul operator ca orice alt operator, atunci construcţia:

$$a.b.c.d \qquad (7.11)$$

este o expresie care se evaluează ca orice altă expresie aritmetică, logică sau raţională. O astfel de expresie o numim *expresie referenţială*.

Expresia referenţială se evaluează de la stânga la dreapta, pentru că în final trebuie să se obţină o adresă. Referirea oricărui operand se face fie prin numele său, dacă abordarea este la nivel de limbaj, fie prin adresă, dacă analiza este din punctul de vedere al programatorului interesat să cunoască mecanismele proprii execuţiei programelor.

Interpretarea expresiei 7.11 este:

- *d* este membru în structura de tip articol *c*
- *c* este membru în structura de tip articol *b*
- *b* este membru în structura de tip articol *a*

ceea ce ca model grafic îi corespunde:

$$
\begin{aligned}
&adr(a.b.c.d) = adr(a.b) + adr(b.c) + adr(c.d) - 2*adr(a) \\
&adr(a.b.c.d) = adr(a) + adr(a.b) + adr(b.c) + adr(c.d) - 3*adr(a) \\
&adr(a.b.c.d) = adr(a) + [adr(a.b) - adr(a)] + [adr(b.c) - \\
&adr(a)]+[adr(c.d)-adr(a)] \qquad (7.12)
\end{aligned}
$$

Punerea corect în corespondenţă a baiţilor ocupaţi de o structură de tip articol cu câmpurile acesteia, mai ales în cazul în care există diferenţe

generate de alocarea neoptimizată a memoriei şi de apariţia variabilelor θ_k, permite interpretarea riguroasă a conţinutului fiecărui câmp. Problematica devine cu atât mai importantă cu cât variabilele de tip articol se utilizează pentru partiţionarea memoriei alocate unor buffere utilizate în operaţii de intrare/ieşire.

Neoptimizarea prelucrărilor este benefică cu condiţia ca interpretarea grupului de baiţi să fie controlată de programator, chiar dacă citirea lor are un alt şablon decât cel utilizat la scriere.

7.3 Vectori de structuri şi structuri de vectori

Vectorii de structurii sunt masive omogene, în sensul că fiecare element al masivului nu diferă de celălalt, chiar dacă în alcătuirea lor intră câmpuri de naturi diferite.

Construcţia:

```
typedef  struct student
{
      char nume[30];
      int varsta;
      char facult[20];
      int an_studiu;
};
student x[20];
```

defineşte un vector *x* având 20 de componente; fiecare componentă este un articol ce conţine câmpurile *nume, facult, an_studiu*. Modelul grafic al acestui tip de dată derivat este reprezentat în figura 7.3.

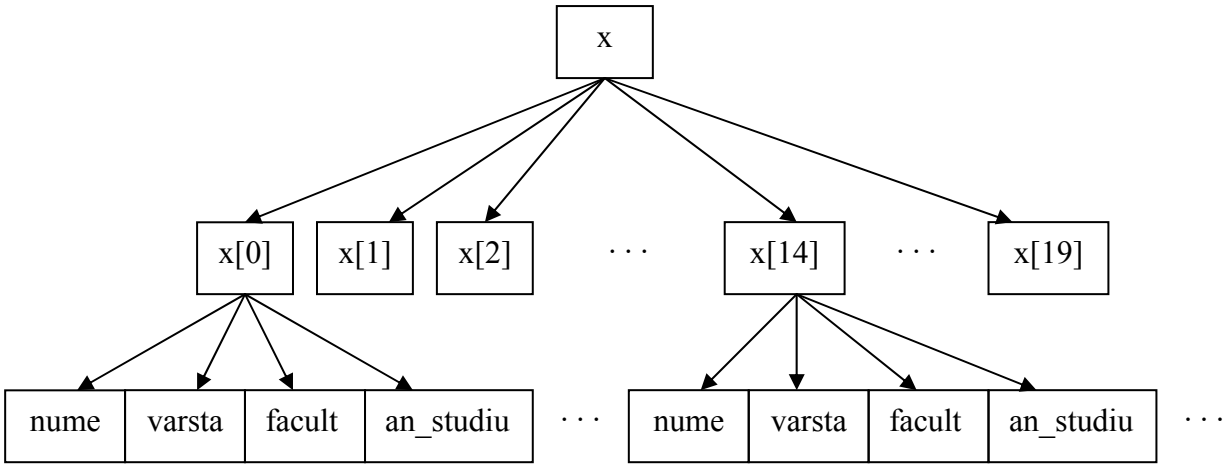

Figura 7.3 *Modelul grafic asociat structurii vectorului de articole x*

Masivul *x* este unul unidimensional omogen, pentru că toate cele 20 de componente au aceeaşi structură.

Referirea vârstei corespunzătoare studentului al cincisprezecelea dintr-o grupă se realizează prin:

```
x[14].varsta
```

În cazul în care câmpurile unui articol sunt elemente ale unui masiv unidimensional, definim o structură de vectori. De exemplu:

```
typedef  struct student
{
      char nume[30];
      int note[10];
};
student x;
```

Dacă se doreşte aflarea notei a 4-a a studentului *x*, referirea se efectuează prin:

```
x.nota[3]
```

Se definesc vectori de structuri care conţin vectori, respectiv vectori de structuri de vectori:

```
stud y[20];
```

Pentru a referi nota a 5-a a studentului al 17-lea se face referirea:

```
y[16].nota[4]
```

Lucrurile iau amploare dacă se definesc structuri de matrice şi matrice de structuri.

Astfel, dacă într-o secţie sunt 20 de muncitori şi întreprinderea are 30 de secţii şi pentru fiecare muncitor trebuie cunoscut: timpul lucrat, salariul orar, numele, definind:

```
typedef struct muncitor
{
      char nume[20];
      long int salariu;
      int ore;
 };
long int salariu_total;
muncitor muncit[30][20];
```

se calculează salariul total pentru muncitorul 15 din secţia a 4-a astfel:

*salariu_total: = muncit[3][14].salariu*muncit[3][14].ore;* *(7.13)*

Dacă matricea este privită ca vectori de vectori, extensiile sunt făcute pentru masivele tridimensionale ca vectori de matrice sau matrice de vectori, iar pentru masivele cu patru dimensiuni, ca masive bidimensionale de masive bidimensionale, sau vectori de masive tridimensionale sau masive tridimensionale de vectori.

Generalizările decurg din posibilitatea de a construi structuri de structuri şi uneori de a introduce aspectul recursiv al descrierii. Secvenţa:

```
typedef int a[5];
typedef a b[5];
b c[5];
```

corespunde descrierii:

```
int c[5][5][5];
```

iar:

```
typedef int x[5][5];
x y[5][5];
```

corespunde descrierii:

```
int z[5][5][5][5];
```

Referirea elementelor se efectuează după aceleaşi reguli, specificând valori între paranteze pătrate, în număr egal cu dimensiunea atribuită masivului, ca de exemplu:

c[i][j][k] *(7.14)*

y[i][j][k][h] *(7.15)*

Atributele sunt comutative în raport cu operatorul de referire. Un element din vectorul de structură se referă prin:

nume_vector[i].nume_membru *(7.16)*

iar un element din structura de vectori se referă prin:

nume_structură.nume_membru[i] *(7.17)*

Se consideră spre exemplificare următorul program care gestionează acţionarii unei societăţi comerciale.

```
#include <windows.h>
#include <process.h>
#include <ctype.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>

struct data
{
      int  da_year,da_day,da_mon;
};

struct adresa
{
```

```
	char str[25];
	int  nr;
	char  bloc[3];
	char  sc;
	int ap;
	int sect;
	long codp;
	char loc[25];
	char jud[20];
};

struct ID
{
	char seria[3];
	long nr;
	data dataE;
	int circa;
};

struct actionar
{
	char cod[6];
	char nume[30];
	struct adresa adr;
	data dataN;
	struct ID BI;
	char reprez[6];
	int nr_act;
};

struct ind
{
	char cheie[6];
	long poz;
};

struct nod
{
	char sir[6];
	nod *st;
	nod *dr;
};

ind index[1000];

char *mes[40]={
	"Data trebuie sa fie numerica!",
		"Ziua trebuie sa fie intre 1 si 31!",
		"Luna trebuie sa fie intre 1 si 12!",
		"Anul trebuie sa fie intre 1900 si 2001!",
		"",
		"Data trebuie sa fie cuprinsa intre 0 si 1!",
		"Serie incorecta!"
};

long lg_fis()
{
	long lung;
	FILE *pf;
	pf=fopen("actionar.dat","rb");
	fseek(pf, 0, SEEK_END);
```

```
      lung=ftell(pf);
      fclose(pf);
      return lung;
}

void insnod(nod *& rad,char *s,int &vb)
{
      if(rad==NULL)
      {
            rad=new nod;
            strncpy(rad->sir,s,6);
            rad->st=rad->dr=NULL;
      }
      else if(strncmp(s,rad->sir,6) < 0) insnod(rad->st,s,vb);
      else if(strncmp(s,rad->sir,6) > 0) insnod(rad->dr,s,vb);
      else vb=1;
}

void del_arb(struct nod *T)
{
      if (T!=NULL)
      {
            del_arb(T->st);
            del_arb(T->dr);
            delete T;
      }
}

int indexare()
{
      FILE *pf;
      actionar act;

      if ((pf=fopen("actionar.dat","rb"))==NULL)
      {
            printf("Fiserul cu date despre actionari nu exista!");
            getch();
            exit(1);
      }
      int k=0;
      long pozitie;
      while(fread(&act,sizeof(struct actionar),1,pf))
      {
            pozitie=ftell(pf)-sizeof(struct actionar);
            strncpy(index[k].cheie,act.cod,6);
            index[k].poz=pozitie;
            k++;
      }
      fclose(pf);
      return k;

}

void sortare(int n)
{
      struct ind temp;
      for(int i=0;i<n-1;i++)
            for(int j=i;j<n;j++)
                  if(strcmp(index[i].cheie,index[j].cheie)>0)
                  {
                        temp=index[i];
```

```
                        index[i]=index[j];
                        index[j]=temp;
                  }
}

int cautare(int n,int &poz,char *key)
{
      int s,m,d;
      s=0;
      d=n;
      while(s<=d)
      {
            m=(s+d)/2;
            if (!strncmp(key,index[m].cheie,6))
            {
                  poz=m;
                  return m;
            }
            else
                  if (strncmp(key,index[m].cheie,6)<0)
                        d=m-1;
                  else
                        s=m+1;
      }
      return -1;
}


void sortareN()
{
      struct actionar act,act1;
      FILE *pf,*pt;
      if((pf=fopen("actionar.dat","rb+"))==NULL)
      {
            printf("Fisierul cu date despre actionari nu exista!");
            getch();
            return;
      }
      pt=fopen("temp.dat","wb");
      int vb,na=lg_fis()/sizeof(struct actionar);
      for(int i=0;i<na;i++)
      {
            fread(&act,sizeof(struct actionar),1,pf);
            fwrite(&act,sizeof(struct actionar),1,pt);
      }
      fclose(pt);
      fclose(pf);
      pt=fopen("temp.dat","rb+");
      do
      {
            vb=0;
            fseek(pt,0,0);
            for(int i=0;i<na-1;i++)
            {
                  fread(&act,sizeof(struct actionar),1,pt);
                  fread(&act1,sizeof(struct actionar),1,pt);
                  if (stricmp(act.nume,act1.nume)>0 )
                  {
                        fseek(pt,ftell(pt)-2*sizeof(struct
actionar),0);
                        fwrite(&act1,sizeof(struct actionar),1,pt);
```

```
                        fwrite(&act,sizeof(struct actionar),1,pt);
                        vb=1;
                  }
                  fseek(pt,ftell(pt)-sizeof(struct actionar),0);
            }
      }
      while(vb==1);
      fclose(pt);
}

void eroare(int cod)
{
      printf("\a%s",mes[cod-1]);
      Sleep(500);
}

void valid_nr_l(long &val,int cod)
{
      int i,VB=0;
      do
      {
            VB=0;

            char tab[255];
            fflush(stdin);
            gets(tab);
            if (strlen(tab)==0)
                  VB=1;
            else
            {
                  i=0;
                  while(VB==0&&i<strlen(tab))
                  {
                        if(isdigit(tab[i]))
                              i++;
                        else
                              VB=1;
                  }
            }
            if(VB==0)
                  sscanf(tab,"%ld",&val);
            else
                  eroare(cod);
      }
      while(VB!=0);
}

void valid_nr_f(float &val,int cod)
{
      int i,VB=0;
      do
      {
            VB=0;

            char tab[255];
            fflush(stdin);
            gets(tab);
            if (strlen(tab)==0)
                  VB=1;
            else
            {
```

```
                  i=0;
                  while(VB==0&&i<strlen(tab))
                  {
                        if(isdigit(tab[i]))
                              i++;
                        else
                              VB=1;
                  }
            }
            if(VB==0)
                  sscanf(tab,"%f",&val);
            else
                  eroare(cod);
      }
      while(VB!=0);
}

void valid_nr_i(int &val,int cod)
{
      int i,VB=0;
      do
      {
            VB=0;

            char tab[255];
            fflush(stdin);
            gets(tab);
            if (strlen(tab)==0)
                  VB=1;
            else
            {
                  i=0;
                  while(VB==0&&i<strlen(tab))
                  {
                        if(isdigit(tab[i]))
                              i++;
                        else
                              VB=1;
                  }
            }
            if(VB==0)
                  sscanf(tab,"%d",&val);
            else
                  eroare(cod);
      }
      while(VB!=0);
}

void valid_ab_sb(char val[],int cod)
{
      int i,VB=0;
      do
      {
            VB=0;
            char tab[255];
            fflush(stdin);
            gets(tab);
            if (strlen(tab)==0||strlen(tab)!=2)
                  VB=1;
            else
            {
```

```
                  i=0;
                  while(VB==0&&i<2)
                  {
                        if(isalpha(tab[i]))
                              i++;
                        else
                              VB=1;
                  }
            }
            if(VB==0)
                  strncpy(val,tab,3);
            else
                  eroare(cod);
      }
      while(VB!=0);
}

void sortareA()
{
      struct actionar act,act1;
      FILE *pf,*pt;
      if((pf=fopen("actionar.dat","rb+"))==NULL)
      {
            printf("Fisierul cu date despre actionari nu exista!");
            getch();
            return;
      }
      pt=fopen("temp.dat","wb");
      int vb,na=lg_fis()/sizeof(struct actionar);
      for(int i=0;i<na;i++)
      {
            fread(&act,sizeof(struct actionar),1,pf);
            fwrite(&act,sizeof(struct actionar),1,pt);
      }
      fclose(pt);
      fclose(pf);
      pt=fopen("temp.dat","rb+");
      do
      {
            vb=0;
            fseek(pt,0,0);
            for(int i=0;i<na-1;i++)
            {
                  fread(&act,sizeof(struct actionar),1,pt);
                  fread(&act1,sizeof(struct actionar),1,pt);
                  if ((unsigned)act.nr_act<(unsigned)act1.nr_act)
                  {
                        fseek(pt,ftell(pt)-2*sizeof(struct
actionar),0);
                        fwrite(&act1,sizeof(struct actionar),1,pt);
                        fwrite(&act,sizeof(struct actionar),1,pt);
                        vb=1;
                  }
                  fseek(pt,ftell(pt)-sizeof(struct actionar),0);
            }
      }
      while(vb==1);
      fclose(pt);
}
void creare()
{
```

```
      int v;
      char c,key[6];
      FILE *pf;
      actionar act;
      nod *RAD=NULL;
      if ((pf=fopen("actionar.dat","rb"))!=NULL)
      {
            printf("ATENTIE\n");
            printf("Fiserul cu date despre actionari exista!\n");
            printf("Datele existente se vor pierde!\n");
            printf("Doriti sa-l rescrieti?[D/N]: ");
            do
            c=getch();
            while(c!='D'&&c!='N');
            if (c=='N')
            {
                  return;
            }
      }
      if ((pf=fopen("actionar.dat","wb"))==NULL)
      {
            printf("Eroare la creare!");
            exit(1);
      }
      else
      {
            printf("Cod: ");
            while(scanf("%6s",key)!=0)
            {
                  v=0;
                  insnod(RAD,key,v);
                  if (v)
                  {
                        printf("\n\aArticolul    cu    aceasta    cheie
exista!");
                        Sleep(500);
                        fflush(stdin);
                        continue;
                  }
                  strncpy(act.cod,key,6);
                  fflush(stdin);
                  printf("Nume: ");
                  gets(act.nume);
                  fflush(stdin);
                  printf("Adresa actionarului\n");
                  printf("\tStrada: ");
                  gets(act.adr.str);
                  printf("\tNumarul: ");
                  valid_nr_i(act.adr.nr,1);
                  printf("\tBloc: ");
                  scanf("%s",act.adr.bloc);
                  fflush(stdin);
                  printf("\tScara: ");
                  scanf("%c",&act.adr.sc);
                  printf("\tApartament: ");

                  valid_nr_i(act.adr.ap,1);
                  printf("\tSector: ");
                  valid_nr_i(act.adr.sect,1);
                  printf("\tCod postal: ");
                  valid_nr_l(act.adr.codp,1);
```

```
                fflush(stdin);
                printf("\tLocalitatea: ");
                gets(act.adr.loc);
                printf("\tJudetul: ");
                gets(act.adr.jud);
                do
                {
                      printf("Data nasterii\n");
                      printf("\tZiua:");
                      valid_nr_i(act.dataN.da_day,1);
                      if(act.dataN.da_day<1||act.dataN.da_day>31)
                            eroare(2);
                }
                while(act.dataN.da_day<1||act.dataN.da_day>31);
                printf("\tLuna:");
                do
                {
                      valid_nr_i(act.dataN.da_mon,1);
                      if(act.dataN.da_mon<1||act.dataN.da_mon>12)
                            eroare(3);
                }
                while(act.dataN.da_mon<1||act.dataN.da_mon>12);
                printf("\tAnul:");
                do
                {
                      valid_nr_i(act.dataN.da_year,1);

    if(act.dataN.da_year<1900||act.dataN.da_year>2001)
                            eroare(4);
                }
    while(act.dataN.da_year<1900||act.dataN.da_year>2001);
                printf("Buletin de identitate\n");
                printf("\tSeria:");
                valid_ab_sb(act.BI.seria,7);
                printf("\tNumarul:");
                valid_nr_l(act.BI.nr,1);
                printf("Data eliberarii\n");
                printf("\tZiua:");

                do
                {
                      valid_nr_i(act.BI.dataE.da_day,1);

    if(act.BI.dataE.da_day<1||act.BI.dataE.da_day>31)
                            eroare(2);
                }
                while(act.BI.dataE.da_day<1||act.BI.dataE.da_day>31);

                printf("\tLuna:");
                do
                {
                      valid_nr_i(act.BI.dataE.da_mon,1);

    if(act.BI.dataE.da_mon<1||act.BI.dataE.da_mon>12)
                            eroare(3);
                }
                while(act.BI.dataE.da_mon<1||act.BI.dataE.da_mon>12);

                printf("\tAnul: ");
                do
                {
```

```
                        valid_nr_i(act.BI.dataE.da_year,1);

      if(act.BI.dataE.da_year<1900||act.BI.dataE.da_year>2001)
                              eroare(4);
                  }

      while(act.BI.dataE.da_year<1900||act.BI.dataE.da_year>2001);
                  printf("\tCirca de politie: ");

                  valid_nr_i(act.BI.circa,1);
                  printf("Numarul de actiuni detinute: ");

                  valid_nr_i(act.nr_act,1);

                  printf("\n");
                  fwrite(&act,sizeof(struct actionar),1,pf);
                  printf("Cod: ");
            }
      }
      del_arb(RAD);
      fclose(pf);
}

void creare_date()
{
      char c;
      float ksoc,pfn,cota;
      long int val_A;
      FILE *pd;
      if ((pd=fopen("date.dat","rb"))!=NULL)
      {
            printf("ATENTIE\n");
            printf("Fiserul cu date despre profit exista!\n");
            printf("Datele existente se vor pierde!\n");
            printf("Doriti sa-l rescrieti?[D/N]: ");
            do
            c=getch();
            while(c!='D'&&c!='N');
            if (c=='N')
            {
                  return;
            }
      }
      if ((pd=fopen("date.dat","wb"))==NULL)
      {
            printf("Eroare la creare!");
            getch();
            return;
      }
      printf("DATE PRIVIND CAPITALUL SOCIAL SI PROFITUL\n");
      printf("Valoarea unei actiuni:");
      valid_nr_l(val_A,1);
      fwrite(&val_A,sizeof(long int),1,pd);
      printf("Capital Social:");
      valid_nr_f(ksoc,1);
      fwrite(&ksoc,sizeof(float),1,pd);
      printf("Profit net:");
      valid_nr_f(pfn,1);
      fwrite(&pfn,sizeof(float),1,pd);
      do
      {
```

```
            printf("Cota din profit repartizata la dividende
[subunitara]:");
            scanf("%f",&cota);
            if (cota>1||cota <0)
                  eroare(6);
      }
      while(cota>1||cota <0);
      fwrite(&cota,sizeof(float),1,pd);
      fclose(pd);
}

void afis_date(const actionar &act)
{
      printf("\n\nCod: %s\n",act.cod);
      printf("Nume: "); puts(act.nume);
      printf("Adresa actionarului\n");
      printf("\tStrada:"); puts(act.adr.str);
      printf("\tNumarul: %d\n",act.adr.nr);
      printf("\tBloc: %s ",act.adr.bloc);
      printf("Scara:       %c       Apartament:       %d       Sector:
%d\n",act.adr.sc,act.adr.ap,act.adr.sect);
      printf("\tCod postal: %ld\n",act.adr.codp);
      printf("\tLocalitatea: ");puts(act.adr.loc);
      printf("\tJudetul: "); puts(act.adr.jud);
      printf("Data                                                nasterii:
%d/%d/%d\n",act.dataN.da_day,act.dataN.da_mon,act.dataN.da_year);
      printf("Buletin de identitate\n");
      printf("\tSeria: %s",act.BI.seria);
      printf(" Numarul: %ld\n",act.BI.nr);
      printf("\tData                                          eliberarii:
%d/%d/%d\n",act.BI.dataE.da_day,act.BI.dataE.da_mon,act.BI.dataE.da_ye
ar);
      printf("\tCirca de politie: %d\n",act.BI.circa);

      printf("Numarul de actiuni detinute: %u\n",act.nr_act);
      printf("\n");

}

void cons()
{
      int n;
      char key[6];
      int vb,i,c;
      actionar act;
      n=indexare();
      sortare(n);
      FILE *pf=fopen("actionar.dat","rb");
      while(printf("\nCod actionar:"),(c=scanf("%6s",key))!=0)
      {
            vb=cautare(n,i,key);
            if(vb==-1)
            {
            printf("\n\aArticolul cu aceasta cheie nu exista!\n");
            Sleep(500);
            fflush(stdin);
            }
            else
            {
                  fseek(pf,index[i].poz,0);
                  fread(&act,sizeof(struct actionar),1,pf);
```

```
                  afis_date(act);
                  getch();
            }
      }
      fclose(pf);
}

void modif()
{
      int n;
      char key[6];
      int vb,i,nr;
      actionar act;
      n=indexare();
      sortare(n);
      FILE *pf=fopen("actionar.dat","rb+");
      while(printf("\nCod actionar:"),scanf("%6s",key)!=EOF)
      {
            vb=cautare(n,i,key);
            if(vb==-1)
            {
                  printf("\n\aArticolul cu aceasta cheie nu exista!");
                  Sleep(500);
            }
            else
            {
                  fseek(pf,index[i].poz,0);
                  fread(&act,sizeof(struct actionar),1,pf);
                  afis_date(act);
                  printf("Noul numar:");
                  valid_nr_i(nr,1);
                  act.nr_act=nr;
                  fseek(pf,index[i].poz,0);
                  fwrite(&act,sizeof(struct actionar),1,pf);
            }
      }
      fclose(pf);
}

void listA()
{
      FILE *pt,*ptxt;
      unsigned int tot_a=0;
      long int val_a;
      float ksoc;
      actionar act;
      sortareA();
      FILE *pd=fopen("date.dat","rb");
      fread(&val_a,sizeof(long int),1,pd);
      fread(&ksoc,sizeof(float),1,pd);
      fclose(pd);
      pt=fopen("temp.dat","rb");
      while(fread(&act,sizeof(struct actionar),1,pt))
            tot_a+=act.nr_act;
      rewind(pt);
      ptxt=fopen("lista1.txt","w");
      fprintf(ptxt,"\n\n\n\t\tLista actionarilor dupa numarul de
actiuni\n");
      fprintf(ptxt,"\t\t*****************************************\n\n
");
```

```
      fprintf(ptxt,"**************************************************
**********\n");
      fprintf(ptxt,"* Cod   *    Nume       Prenume          *  Numar  *
Ponderea actiunilor*\n");
      fprintf(ptxt,"*       *                                  * actiuni *
in capitalul social*\n");
      fprintf(ptxt,"**************************************************
**********\n");
      while(fread(&act,sizeof(struct actionar),1,pt))
      {
            fprintf(ptxt,"*%6s            *%-30s*%8u            *%18.3f
%%*\n",act.cod,act.nume,act.nr_act,(float)(unsigned)act.nr_act*val_a/k
soc*100);

      fprintf(ptxt,"**************************************************
**********\n");
      }
      fprintf(ptxt,"**************************************************
**********\n");
      fclose(pt);
      fclose(ptxt);
}

void listD()
{
      FILE *pt,*ptxt;
      SYSTEMTIME data;
      float pfn,cota;
      float div;
      actionar act;
      sortareA();
      unsigned int tot_a=0;
      FILE *pd=fopen("date.dat","rb");
      fseek(pd,sizeof(long int)+sizeof(float),0);
      fread(&pfn,sizeof(float),1,pd);
      fread(&cota,sizeof(float),1,pd);
      fclose(pd);
      pt=fopen("temp.dat","rb");
      while(fread(&act,sizeof(struct actionar),1,pt))
            tot_a+=act.nr_act;
      div=(pfn*cota)/tot_a;
      rewind(pt);
      ptxt=fopen("lista2.txt","w");
      GetSystemTime(&data);
      fprintf(ptxt,"\n\t\t\t\tDividende cuvenite actionarilor la data
%d.%d.%d\n",data.wDay,data.wMonth,data.wYear);
      fprintf(ptxt,"\t\t\t\t****************************************
******\n\n");
      fprintf(ptxt,"Profit net:%.0f lei\n",pfn);
      fprintf(ptxt,"Cota      de      profit      repartizata      la
dividende:%.2f\n",cota);
      fprintf(ptxt,"Total actiuni:%u\n",tot_a);
      fprintf(ptxt,"Dividende per actiune:%.2f lei/act\n\n",div);
      fprintf(ptxt,"**************************************************
**************************************************************
\n");
      fprintf(ptxt,"*     Nume        Prenume            *   Data     *
Adresa                          *Buletin de *   Numar * Dividende
*\n");
      fprintf(ptxt,"*                                        * nasterii *
*identitate * actiuni*           *\n");
```

```
      fprintf(ptxt,"**************************************************
**************************************************************
\n");
      while(fread(&act,sizeof(struct actionar),1,pt))
      {
            fprintf(ptxt,"*%-30s*%2d.%2d.%d*Str:%-29s      Nr:%-5d*%-2s
%8ld*%8u*%11.2f
*\n",act.nume,act.dataN.da_day,act.dataN.da_mon,act.dataN.da_year,act.
adr.str,act.adr.nr,act.BI.seria,act.BI.nr,act.nr_act,(unsigned)act.nr_
act*div);
            fprintf(ptxt,"*                                                *
*Bl:%c%c%c Sc:%c Ap:%-3d Sect:%-2d Cod p:%-8ld *            *         *
*\n",act.adr.bloc[0],act.adr.bloc[1],act.adr.bloc[2],act.adr.sc,act.ad
r.ap,act.adr.sect,act.adr.codp);
            fprintf(ptxt,"*                                                *
*Loc:%-20s                        *                 *             *
*\n",act.adr.loc);

      fprintf(ptxt,"**************************************************
**************************************************************
\n");
      }
      fprintf(ptxt,"**************************************************
**************************************************************
\n");
      fclose(pt);
      fclose(ptxt);
}

void listAB()
{
      FILE *pt,*ptxt;

      actionar act;

      sortareN();
      pt=fopen("temp.dat","rb");
      ptxt=fopen("lista3.txt","w");
      fprintf(ptxt,"\n\n\n\t\t\tLista     actionarilor     in     ordine
alfabetica\n");
      fprintf(ptxt,"\t\t\t**************************************\n\n"
);
      fprintf(ptxt,"\t\t*********************************************
****\n");
      fprintf(ptxt,"\t\t*   Nr.   *      Nume         Prenume            *
Numar  *\n");
      fprintf(ptxt,"\t\t* crt.   *                                         *
actiuni *\n");
      fprintf(ptxt,"\t\t*********************************************
****\n");
      int i=0;
      while(fread(&act,sizeof(struct actionar),1,pt))
      {
            fprintf(ptxt,"\t\t*%6d                              *%-30s*%8u
*\n",++i,act.nume,act.nr_act);

      fprintf(ptxt,"\t\t*********************************************
****\n");
      }
      fprintf(ptxt,"\t\t*********************************************
****\n");
```

```
	fclose(pt);
	fclose(ptxt);
}

void do_sterg()
{
	int n,c;
	int vb,i;
	actionar act;
	n=indexare();
	sortare(n);
	char key[6];
	FILE *pf=fopen("actionar.dat","rb");
	while(printf("\nCod actionar:"),scanf("%6s",key)!=0)
	{
		vb=cautare(n,i,key);
		if(vb==-1)
		{
		printf("\n\aArticolul cu aceasta cheie nu exista!\n");
		Sleep(500);
		fflush(stdin);
		continue;
		}
		else
		{
			fseek(pf,index[i].poz,0);
			fread(&act,sizeof(struct actionar),1,pf);
			afis_date(act);
			printf("Doriti sa stergeti acest articol?[D/N]: ");
			do
			{
				c=getch();
			}
			while(c!='D'&&c!='N');
			if (c=='N')
			{
				continue;
			}
		}
		//rescriere fisier
		FILE *ptmp=fopen("tmp.dat","wb");
		rewind(pf);
		for(int j=0;j<index[i].poz/sizeof(struct actionar);j++)
		{
			fread(&act,sizeof(struct actionar),1,pf);
			fwrite(&act,sizeof(struct actionar),1,ptmp);
		}
		fread(&act,sizeof(struct actionar),1,pf);
		while(fread(&act,sizeof(struct actionar),1,pf))
			fwrite(&act,sizeof(struct actionar),1,ptmp);
		fclose(pf);
		fclose(ptmp);
		unlink("actionar.dat");
		rename("tmp.dat","actionar.dat");
		n=indexare();
		sortare(n);
		pf=fopen("actionar.dat","rb");
	}

	fclose(pf);
}
```

```
void adaug()
{
      int v;
      char key[6];
      actionar act;
      nod *RAD=NULL;
      FILE *pf;
      if ((pf=fopen("actionar.dat","rb+"))==NULL)
      {
            printf("Fiserul cu date despre actionari nu exista!");
            return;
      }
      else
      {
            strncpy(act.reprez,"      ",6);
            while(fread(&act,sizeof(struct actionar),1,pf))
                  insnod(RAD,act.cod,v);
            //fseek(pf,0,2);
            printf("Cod:\n");
            while(scanf("%6s",key)!=0)
            {
                  v=0;
                  insnod(RAD,key,v);
                  if (v)
                  {

                  printf("\n\aArticolul cu aceasta cheie exista!\n");
                  Sleep(500);
                  fflush(stdin);
                  continue;
                  }
                  strncpy(act.cod,key,6);
                  fflush(stdin);
                  printf("Nume: ");
                  gets(act.nume);
                  fflush(stdin);
                  printf("Adresa actionarului\n");
                  printf("\tStrada: ");
                  gets(act.adr.str);
                  printf("\tNumarul: ");
                  valid_nr_i(act.adr.nr,1);
                  printf("\tBloc: ");
                  scanf("%s",act.adr.bloc);
                  fflush(stdin);
                  printf("\tScara: ");
                  scanf("%c",&act.adr.sc);
                  printf("\tApartament: ");
                  valid_nr_i(act.adr.ap,1);
                  printf("\tSector: ");
                  valid_nr_i(act.adr.sect,1);
                  printf("\tCod postal: ");
                  valid_nr_l(act.adr.codp,1);
                  fflush(stdin);
                  printf("\tLocalitatea: ");
                  gets(act.adr.loc);
                  printf("\tJudetul: ");
                  gets(act.adr.jud);
                  do
                  {
                        printf("Data nasterii\n");
```

```
                        printf("\tZiua:");
                        valid_nr_i(act.dataN.da_day,1);
                        if(act.dataN.da_day<1||act.dataN.da_day>31)
                              eroare(2);
                  }
                  while(act.dataN.da_day<1||act.dataN.da_day>31);

                  printf("\tLuna:");
                  do
                  {
                        valid_nr_i(act.dataN.da_mon,1);
                        if(act.dataN.da_mon<1||act.dataN.da_mon>12)
                              eroare(3);
                  }
                  while(act.dataN.da_mon<1||act.dataN.da_mon>12);

                  printf("\tAnul:");
                  do
                  {
                        valid_nr_i(act.dataN.da_year,1);

      if(act.dataN.da_year<1900||act.dataN.da_year>2001)
                              eroare(4);
                  }
      while(act.dataN.da_year<1900||act.dataN.da_year>2001);
                  printf("Buletin de identitate\n");
                  printf("\tSeria:");
                  valid_ab_sb(act.BI.seria,7);
                  printf("\tNumarul:");
                  valid_nr_l(act.BI.nr,1);
                  printf("Data eliberarii\n");
                  printf("\tZiua:");
                  do
                  {
                        valid_nr_i(act.BI.dataE.da_day,1);

      if(act.BI.dataE.da_day<1||act.BI.dataE.da_day>31)
                              eroare(2);
                  }
                  while(act.BI.dataE.da_day<1||act.BI.dataE.da_day>31);

                  printf("\tLuna:");
                  do
                  {
                        valid_nr_i(act.BI.dataE.da_mon,1);

      if(act.BI.dataE.da_mon<1||act.BI.dataE.da_mon>12)
                              eroare(3);
                  }
                  while(act.BI.dataE.da_mon<1||act.BI.dataE.da_mon>12);
                  printf("\tAnul: ");
                  do
                  {
                        valid_nr_i(act.BI.dataE.da_year,1);

      if(act.BI.dataE.da_year<1900||act.BI.dataE.da_year>2001)
                              eroare(4);
                  }
while(act.BI.dataE.da_year<1900||act.BI.dataE.da_year>2001);
                  printf("\tCirca de politie: ");
```

```
                  valid_nr_i(act.BI.circa,1);
                  printf("Numarul de actiuni detinute: ");

                  valid_nr_i(act.nr_act,1);

                  printf("\n");
                  fwrite(&act,sizeof(struct actionar),1,pf);
                  printf("Cod: ");
            }
      }
      del_arb(RAD);
      fclose(pf);
}

void men_fis()
{
      int rasp,cont=1;
      while(cont)
      {
            do
            {
                  printf("MENIUL FISIERE\n\n");
                  printf("1.CREARE FISIER ACTIONARI\n");
                  printf("2.CREARE FISER CU DATE PRIVIND PROFITUL\n" );
                  printf("3.ADAUGARE ACTIONARI\n");
                  printf("4.MODIFICARE NUMAR DE ACTIUNI\n");
                  printf("5.STERGERE ACTIONARI DIN FISIER\n");
                  printf("6.REVENIRE LA MENIUL PRINCIPAL\n\n");
                  printf("Optiunea:");
                  valid_nr_i(rasp,5);//scanf("%d",&rasp);
            }
            while(rasp<1||rasp>6);
            switch(rasp)
            {
            case 1:creare();break;
            case 2:creare_date();break;
            case 3:adaug();break;
            case 4:modif();break;
            case 5:do_sterg();break;
            case 6:cont=0;break;
            }
      }
}

void men_sit()
{
      int rasp,cont=1;
      while(cont)
      {
            do
            {
            printf("MENIUL SITUATII DE IESIRE\n\n");
            printf("1.LISTA ACTIONARILOR IN ORDINEA NUMARULUI DE
ACTIUNI\n");
                  printf("2.LISTA ACTIONARILOR IN ORDINE
ALFABETICA\n");
                  printf("3.LISTA DIVIDENDELOR CUVENITE FIECARUI
ACTIONAR\n");
                  printf("4.CONSULTARE FISIER DUPA COD ACTIONAR\n");
                  printf("5.REVENIRE LA MENIUL PRINCIPAL\n\n");
                  printf("Optiunea:");
```

```
                valid_nr_i(rasp,5);//scanf("%d",&rasp);
            }
            while(rasp<1||rasp>5);
            switch(rasp)
            {
            case 1:listA();break;
            case 2:listAB();break;
            case 3:listD();break;
            case 4:cons();break;
            case 5:cont=0;break;
            }
      }
}

void main()
{
      int rasp,cont=1;
      while(cont)
      {
            do
            {
                  printf("MENIUL PRINCIPAL\n\n");
                  printf("1.FISIERE\n");
                  printf("2.SITUATII DE IESIRE\n");
                  printf("3.TERMINARE PROGRAM\n\n");
                  printf("Optiunea:");
                  valid_nr_i(rasp,5);//scanf("%d",&rasp);
            }
            while(rasp<1||rasp>3);
            switch(rasp)
            {
            case 1:men_fis();break;
            case 2:men_sit();break;
            case 3:cont=0;break;
            }
      }
}
```

Programatorul trebuie să realizeze un echilibru între creşterea numărului de dimensiuni, reducerea gradului de umplere şi complexitatea expresiilor asociate calculelor de deplasare, pentru a localiza fiecare element al structurii pe care o defineşte. Acest echilibru conduce în final la reducerea duratei de prelucrare şi la obţinerea lizibilităţii bune a programului.