Laborator 7
Tablouri de structuri.Uniuni, Enumerări,Câmpuri de Biți

În acest capitol sunt prezentate consideraţii teoretice şi probleme rezolvate privind definirea, utilizarea şi sortarea tablourilor de stucturi.

CONSIDERAŢII TEORETICE

Tablourile de structuri se definesc astfel:

- se defineşte mai întâi un tip de structură
- se declară o variabilă tablou de structuri de tipul structurii definite anterior

Indicii tablourilor de structuri sunt inițializați implicit cu 0, similar cu indicii tablourilor de date standard.

Ex.: definirea unui tablou unidimensional de structuri :

```
#include <stdio.h>
void main()
{struct adrese
        { char nume[30];
        char strada[40];
        char oras[20];
        int cod; } ;
//definire tablou de structuri cu 100 elemente de tip adrese
struct adrese adr[100];
adr[2].cod=3400;
printf("cod=%d",adr[2].cod);}
```

Sortarea tablourilor de structuri se poate realiza utilizând orice metodă de sortare dintre cele utilizate la sortarea tablourilor de date standard (cu elemente intregi, reale, caracter, etc). Cele mai des utilizate metode de sortare sunt:

- Metoda Bublle Sort ,
- Metoda Quick Sort,
- Metoda selectiei,
- Metoda insertiei, etc.

In continuare sunt prezentate modul de declarare si utilizare a uniunilor, enumerarilor, campurilor de biti si altor tipuri de date definite de utilizator cu typedef .

I) Uniunea: este prin definiție o variabilă care poata păstra , la momente diferite , obiecte de tipuri și mărimi diferite. Aceasta variabilă va ocupa suficientă memorie ca să poată stoca cel mai mare dintre tipurile enumerate în componența ei.

Declararea unor **variabile de tip uniune** se poate realiza în două moduri:

a) în două etape:

- Se definește mai întâi tipul uniune cu componentele (elementele) uniunii

- Se declară variabilele de tipul declarat anterior

b)într-o singură etapă:
- Se declară și tipul uniunii și variabilele de acest tip într-o singură instrucțiune.

Declararea tipului uniune se poate realiza astfel:

```
union nume_tip_uniune
                { tip var1;
                  tip var2;
                  …
                 } ;
```

Declararea variabilelor de tip uniune se poate face cu instrucțiunea:

```
union nume_tip_uniune lista_variabile_uniune;
```

Declararea variabilelor de tip uniune într-o singură instrucțiune:

```
union nume_tip_uniune
        { tip var1;
          tip var2;
          …
         } lista_variabile_uniune;
```

Alocarea memoriei se realizează în mod diferit pentru structuri și uniuni. Astfel pentru structuri: zona de memorie ocupată este mai mare sau egală cu suma mărimilor elementelor sale (vezi Exemplu a.). Pentru uniuni: zona de memorie ocupată este egală cu mărimea celui mai mare membru al său (Exemplu b.).

Exemplu a. : Determinarea mărimii zonei de memorie ocupate de structura x

```
struct s
 {   char ch;     //1 octet
     int i;       //2 octeti
     float f;     //4 octeti
 } x;
// sizeof (x) >=7 (=1+2+4)
```

Exemplu b. : Determinarea dimensiunii zonei de memorie ocupate de uniunea, y:

```
union u
 {   char ch;  //1 octet
     int i;  //2 octeti
              float f;      //4 octeti
 } y;
// sizeof (y) =4
```

Accesul la elementele uniunii se realizează similar cu accesul la elementele unei structuri:
- **cu operatorul punct (.)** : nume_uniune.member
- **cu operatorul sageată (->)** : pointer_uniune-> member

Exemplu: acces la elementele unei uniuni

```
union tip
        { int i;
          char ch;
        } cont;
union tip *p;
cont.i=10;
p->i=10;
```

II. Enumerare: este prin definiție un set de constante care specifică toate valorile pe care le poate lua variabila de acel tip.

Declarare variabilă de tip enumerare: similar cu declararea structurilor

enum [nume_enum] { lista enum } lista_variabile_enum ;

typedef enum [nume_enum] {lista enum} nume_tip;

- **nume_ enum** este numele noului tip de date utilizator și este opțional
- **lista enum** este considerată listă de constante de tip întreg, primul element din listă are valoarea 0, al doilea valoarea 1, ș.a.m.d. , dacă nu se inițializează cu alte valori
- **lista_variabile_enum** este lista variabilelor de tipul nume_enum

În acest mod se declară variabile de tipul enumerare, tipul enumerare permițând definirea unei liste de constante întregi cu nume în vederea folosirii de nume sugestive pentru valori numerice.

Exemplu.1: declarare variabilă tip enumerare numită bani de tip monede

```
enum monede {dolar,marca,leu,yen,forint } ;
//declarare tip enumerare
enum monede bani;
//declarare variabila enumerare
```

Exemplu 2: declarare variabilă tip enumerare numită logic de tip Boolean

```
enum Boolean {false, true} logic;
//false=0, true=1,
//se poate utiliza in expresii conditionale: logic==false sau logic == true
```

Exemplu 3: declarare variabilă tip enumerare fără specificarea nume_enum

```
enum { ileg,ian,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna ;
//expresii echivalente: luna =3;  luna=mar; (pentru ca mar=3)
//sau
enum {ian=1,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna;
```

Accesul la elementele enumerării se poate realiza direct, utilizând numele și numărul de ordine din lista de enumerare .

Exemplu : declarare variabila tip enumerare numită bani de tip monede

```
enum monede
        {dolar,marca,leu,yen,forint } ; //declarare tip  enumerare
```

```
enum monede bani;   //declarare variabila enumerare

//instructiuni permise
bani=leu;   //echivalent cu bani =2 pentru ca leu=2
if (bani==forint) printf("Banul este un forint")
printf("%d, %d", dolar,leu);   // va tipari valorile 0,2
```

Inițializarea variabilelor de tip enumerare

Implicit elementele din lista enum sunt inițializate cu valori pornind de la 0,1,… Inițializarea elementelor cu alte valori decât cele implicite se face utilizând semnul egal urmat de o valoare întreagă, modificându-se și valorile elementelor ce urmează după valoarea inițializată

Exemplu : inițializare elemente enumerare:

```
enum monede
        {dolar,marca,leu=100,yen,forint } ;
enum monede bani;
printf("%d, %d, %d,%d,%d", dolar,marca,leu,yen,forint);
// va tipari valorile 0,1,100,101,102
```

Elementele din lista de enumerare nu sunt șiruri de caractere ci sunt o etichetă pentru valori întregi.

Operatorul **typedef** permite definirea unor tipuri particulare definite de utilizatori.

Formatul de definire a unui tip nou de date este : **typedef tip nume_nou**

unde **tip** = orice tip de date existent
nume_nou = numele nou dat tipului tip

Exemplu : declarație de tip float

```
typedef float bilant; //bilant este un alt nume pentru tipul float
bilant scadent;       //se declara variabila scadent de tipul bilant adica float
typedef bilant total;// total  este un alt nume pentru tipul bilant adica
                        //pentru tipul float;
```

Nu se creează de fapt nici un tip nou de date , ci numai un nou nume pentru un tip de date existent.

III. CÂMPURI DE BIȚI

Câmpul de biți: este un element al unei structuri care cuprinde unul sau mai multi biți adiacenți. Câmpurile de biți se pot accesa prin nume, unul sau mai multi biți dintr-un octet sau cuvânt și se pot grupa formând o structură .

Formatul de declarare a unui câmp de biți este :

```
struct nume_struct {
        tip nume1: lungime;
        tip nume2: lungime;
        …
```

tip nume N: lungime;
} lista_variabile;

unde **tip** = tipul câmpului de biți ce poate fi int,unsigned sau signed.
lungime = nr. de biți dintr-un câmp

Câmpul de biți permite accesul la un singur bit. Câmpul de biți cu lungimea 1 trebuie declarat de tip unsigned pentru că un singur bit nu poate avea semn. Câmpurile de biți sunt utilizate frecvent pentru analiza intrării de la un echipament hardware

Restricții de utilizare a variabilelor de tip câmp de biți:

- Nu se poate obține adresa unui câmp de biți cu operatorul &
- Nu pot fi utilizate într-o matrice
- Există restricții de rulare de la stânga la dreapta și invers care diferă de la echipament la echipament

Exemplu : câmp de biți utilizat în cadrul unei structuri. Se definește o înregistrare într-o bază de date despre un angajat care folosește numai un octet pentru a păstra 3 informații:

- statutul angajatului,
- dacă a lucrat în luna respectivă si
- impozitul

```
struct angajat {
        struct adr adrese ;
        float salar ;
        unsigned activ:   1     //statut angajat: activ sau intrerupt
        unsigned orar:    1     //plata orara
        unsigned impozit:1      //impozit rezultat
};
```

Accesul la elementele câmpurilor de biți se realizează similar cu accesul la elementele unei structuri utilizând operatorul punct: **nume_struct.nume_camp**

PROBLEME REZOLVATE

Ex.1. Programul definește o variabilă tablou unidimensional de structuri, și realizează următoarele operații:

- ***citește numărul studenților (dimensiunea n a tabloului), numele, prenumele și 2 note pentru fiecare student***
- ***calculează media aritmetică a notelor pentru fiecare student***
- ***afișează studenții sortați prin Metoda Bulelor în ordinea descrescătoare a mediilor.***

Varianta in C

```
#include <stdio.h>
#include <stdlib.h>
struct student {
        char name[20];
        char prenume[20];
        double nota1;
        double nota2;
        double media;} s[100], aux[100];
int i=0,n,k;
int  main (void)
{printf("Introduceti numarul de studenti:");
scanf("%d", &n);
for (i=0;i<n;i++)
{       printf("\nIntroduceti numele studentului %d:", i+1);      scanf("%s", s[i].name);
        printf("Introduceti prenumele studentului %d:", i+1); scanf("%s",s[i].prenume);
        printf("Introduceti nota1 a studentului %d:",i+1); scanf("%lf", &s[i].nota1);
        printf("Introduceti nota2 a studentului %d:",i+1); scanf("%lf", &s[i].nota2);}
printf("\nLista studentilor:\n");
for (i=0;i<n;i++)
{s[i].media = (s[i].nota1+s[i].nota2)/2;
printf("%8s %6s, nota1=%5.2lf, nota2=%5.2lf, media= %5.2lf \n", s[i].name, s[i].prenume,s[i].nota1, s[i].nota2,
s[i].media);}
printf("\nStudentii sortati in ordinea descrescatoare a mediilor:\n");
do
{k=0;
for (i=0;i<n-1;i++)
   {    if (s[i].media<s[i+1].media)
                {       aux[i]=s[i] ;
                        s[i]=s[i+1];
                        s[i+1]=aux[i];k=1;}
   }
}
while (k);
for (i=0;i<n;i++)
{printf("%8s %6s, nota1=%5.2lf, nota2=%5.2lf, media=%5.2lf \n", s[i].name, s[i].prenume, s[i].nota1,
s[i].nota2,s[i].media);}
 return 0;
}
```

Varianta in C++

```
#include <iostream>
#include <stdlib.h>
using namespace std;
struct student {
        char name[20];
        char prenume[20];
        double nota1;
        double nota2;
        double media;} s[100], aux[100];
int i=0,n,k;
int  main (void)
{cout<<"Introduceti numarul de studenti:";
cin>>n;
for (i=0;i<n;i++)
{       cout<<"\nIntroduceti numele studentului "<< i+1<<":";   cin>>s[i].name;
        cout<<"Introduceti prenumele studentului "<< i+1<<":"; cin>>s[i].prenume;
        cout<<"Introduceti nota1 a studentulu "<<i+1<<":"; cin>>s[i].nota1;
        cout<<"Introduceti nota2 a studentului "<<i+1<<":"; cin>>s[i].nota2;}
cout<<endl<<"Lista studentilor:"<<endl;
for (i=0;i<n;i++)
{s[i].media = (s[i].nota1+s[i].nota2)/2;
cout<<s[i].name<<", "<<s[i].prenume<<", nota1="<<s[i].nota1<<", nota2="<<s[i].nota2<<", media="<<
s[i].media<<endl;}
cout<<"\nStudentii sortati in ordinea descrescatoare a mediilor:"<<endl;
do
{k=0;
for (i=0;i<n-1;i++)
   {    if (s[i].media<s[i+1].media)
                {       aux[i]=s[i] ;
                        s[i]=s[i+1];
                        s[i+1]=aux[i];k=1;}
   }
}
while (k);
for (i=0;i<n;i++)
{cout<<s[i].name<<", "<<s[i].prenume<<", nota1="<<s[i].nota1<<", nota2="<<s[i].nota2<<", media="<<
s[i].media<<endl;}
 return 0;
}
```

Rezultate: ***Aplicație:***

```
Introduceti numarul de studenti:3

Introduceti numele studentului 1:Pop
Introduceti prenumele studentului 1:Ana
Introduceti nota1 a studentului 1:7.5
Introduceti nota2 a studentului 1:8.5

Introduceti numele studentului 2:Balc
Introduceti prenumele studentului 2:Alex
Introduceti nota1 a studentului 2:9
Introduceti nota2 a studentului 2:10

Introduceti numele studentului 3:Zahir
Introduceti prenumele studentului 3:Abdul
Introduceti nota1 a studentului 3:10
Introduceti nota2 a studentului 3:7.5

Lista studentilor:
     Pop    Ana, nota1= 7.50, nota2= 8.50, media=  8.00
    Balc   Alex, nota1= 9.00, nota2=10.00, media=  9.50
   Zahir  Abdul, nota1=10.00, nota2= 7.50, media=  8.75

Studentii sortati in ordinea descrescatoare a mediilor:
    Balc   Alex, nota1= 9.00, nota2=10.00, media= 9.50
   Zahir  Abdul, nota1=10.00, nota2= 7.50, media= 8.75
     Pop    Ana, nota1= 7.50, nota2= 8.50, media= 8.00
```

Să se realizeze sortarea studenților după medii , doar dacă au media mai mare decât 8.50.

Ex.2. Programul citește 3 date calendaristice de la tastatură care au anul peste 2000 și le memorează într-un tablou unidimensional de structuri. Se vor afișa datele pentru care anul este in intervalul (2004, 2019).

Varianta in C

```
#include <stdio.h>
struct  data  {              //definire variabila  globala de tip structura
        int ziua, luna, an;   };
int main()
{ struct data date[3];
int i;
printf("Introduceti 3 date calendaristice\n");
for( i = 0; i < 3; ++i )
{    printf("\nData calendaristica %d:(zz ll aa )",i+1 );
     scanf("%d %d %d", &date[i].ziua, &date[i].luna, &date[i].an );}
printf("Se verifica daca anul este in intervalul [2004,2019]\n");
for( i = 0; i < 3; ++i )
{    if (date[i].an>04 && date[i].an<19)
              printf("Data calendaristica %d este in interval\n",i+1);}
   return 0;}
```

Varianta in C++

```
#include <iostream>
using namespace std;
struct  data  {              //definire variabila  globala de tip structura
        int ziua, luna, an;   };
int main()
{ struct data date[3];
int i;
cout<<"Introduceti 5 date calendaristice"<<endl;
for( i = 0; i < 3; ++i )
{    cout<<"\nData calendaristica "<<i+1<<":(zz ll aa)";
```

```
    cin>>date[i].ziua>>date[i].luna>>date[i].an;}
cout<<"Se verifica daca anul este in intervalul [2004,2019]"<<endl;
for( i = 0; i < 3; ++i )
{    if (date[i].an>04 && date[i].an<19)
 cout<<"Data calendaristica cu nr. "<<i+1<<" este in interval"<<endl;}
   return 0;}
```

Rezultate:

```
Introduceti 5 date calendaristice

Data calendaristica 1:(zz ll aa)15 03 18

Data calendaristica 2:(zz ll aa)12 02 03

Data calendaristica 3:(zz ll aa)19 12 19
Se verifica daca anul este in intervalul [2004,2019]
Data calendaristica cu nr. 1 este in interval
```

Aplicație:

Să se modifice programul astfel încât să se verifice dacă datele introduse sunt într-un interval citit de la tastatură.

Ex.3. Programul citește nr.de angajați și datele lor: nume,prenume, adresa,cod numeric, salar brut și calculează salarul net și impozitul fiecărui angajat afișând la sfârșit un tabel tip stat de plată.

Varianta in C

```
#include <stdio.h>
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
{   char nume[30];
            char prenume[30];
            struct adrese adr ;
            float salarbrut;
            float impozit;
            float salarnet ;};
struct angaj pers[100];
int i,n;
float sumabrut=0.,sumanet=0.,sumaimp=0.;
printf ("Introduceti nr. de angajati: "); scanf("%d", &n);
for (i=0; i<n;i++){
        printf("Introduceti numele angajat %d:", i+1);
        scanf("%10s", pers[i].nume);
        printf("Introduceti prenume angajat %d:", i+1);
        scanf("%10s", pers[i].prenume);
        printf("Introduceti adresa angajat %d", i+1);
        printf("\nIntroduceti strada: ");
        scanf("%10s", pers[i].adr.strada);
        printf("Introduceti orasul: ");
        scanf("%s", pers[i].adr.oras);
        printf("Introduceti cod: ");
        scanf("%d", &pers[i].adr.cod);
        printf("Introduceti salarul brut: ");
        scanf("%f", &pers[i].salarbrut);   printf("\n");
```

```
        pers[i].impozit=pers[i].salarbrut*0.4;
        pers[i].salarnet=pers[i].salarbrut-pers[i].impozit;
        sumabrut+=pers[i].salarbrut;
        sumanet+=pers[i].salarnet;
        sumaimp+=pers[i].impozit;}
printf("*************************************************************************\n");
printf("              Tabel salarii                              \n");
printf("*************************************************************************\n");
printf("Nr.%-8s%-6s %-14s %10s %10s %10s \n", "Nume", "Prenume", "Adresa", "Salar Brut", "Impozit", "Salar
Net");
for (i=0; i<n;i++)
 { printf("%-2d.%-8s %-6s %4s,%4s,%4d %10.f %10.f %10.f\n",(i+1), pers[i].nume,  pers[i].prenume,
pers[i].adr.strada, pers[i].adr.oras, pers[i].adr.cod, pers[i].salarbrut, pers[i].impozit, pers[i].salarnet);
//atentie linie continuata !
printf("\n");}
printf("%33s %10.f %10.f %10.f\n", "Total : ", sumabrut, sumaimp, sumanet);
 return 0;}
```

Varianta in C++

```
#include <iostream>
using namespace std;
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
{   char nume[30];
            char prenume[30];
            struct adrese adr ;
            float salarbrut;
            float impozit;
            float salarnet ;};
struct angaj pers[100];
int i,n;
float sumabrut=0.,sumanet=0.,sumaimp=0.;
cout <<"Introduceti nr. de angajati: "; cin>>n;
for (i=0; i<n;i++){
        cout<<"Introduceti numele angajat "<<i+1<<":";
        cin>>pers[i].nume;
        cout<<"Introduceti prenume angajat "<<i+1<<":";
        cin>>pers[i].prenume;
        cout<<"Introduceti adresa angajat "<<i+1<<":";
        cout<<"\nIntroduceti strada: ";
        cin>>pers[i].adr.strada;
        cout<<"Introduceti orasul: ";
        cin>> pers[i].adr.oras;
        cout<<"Introduceti cod: ";
        cin>>pers[i].adr.cod;
        cout<<"Introduceti salarul brut: ";
        cin>>pers[i].salarbrut;   cout<<endl;
        pers[i].impozit=pers[i].salarbrut*0.4;
        pers[i].salarnet=pers[i].salarbrut-pers[i].impozit;
        sumabrut+=pers[i].salarbrut;
```

```
        sumanet+=pers[i].salarnet;
        sumaimp+=pers[i].impozit;}
cout<<"*************************************************************************"<<endl;
cout<<"              Tabel salarii                      "<<endl;
cout<<"*************************************************************************"<<endl;
cout<<"Nr. Nume      Prenume      Adresa      Salar      Brut      Impozit      Salar Net"<<endl;
for (i=0; i<n;i++)
 { cout<<(i+1)<<"    "<<pers[i].nume<<"     "<<pers[i].prenume<<"
"<<pers[i].adr.strada<<","<<pers[i].adr.oras<<"    "<<pers[i].adr.cod<<"     "<<pers[i].salarbrut<<"    "<<
pers[i].impozit<<"    "<<pers[i].salarnet;
//atentie linie continuata !
cout<<endl;}
cout<< "                                Total : " <<sumabrut<<"       "<<sumaimp<<"       "<<sumanet<<endl;
 return 0;}
```

Rezultate:

```
Introduceti nr. de angajati: 2
Introduceti numele angajat 1:Popa
Introduceti prenume angajat 1:Ioan
Introduceti adresa angajat 1
Introduceti strada: Cioran
Introduceti orasul: Cluj
Introduceti cod: 40020
Introduceti salarul brut: 4000

Introduceti numele angajat 2:Rusu
Introduceti prenume angajat 2:Anca
Introduceti adresa angajat 2
Introduceti strada: Cioran
Introduceti orasul: Cluj
Introduceti cod: 40020
Introduceti salarul brut: 6000
```

Aplicație:

Să se modifice programul astfel încât să se realizeze sortarea în ordine alfabetică a angajaților după nume, sau oraș .

Ex.4. Programul citește numele, prenumele,data nașterii, codul personal pentru n persoane, calculează vârsta fiecărei persoane și apoi afișează toate datele, utilizând alocarea dinamică a memoriei.

Varianta in C

```
#include <stdio.h>
#include <malloc.h>
struct data {
        int ziua; //definitie globala de tip
        int luna;
        int an; };
struct pers {
char nume[15];
          char prenume[20];
          int cod;
          struct data datan;
          int varsta; };
struct pers p[20],*dp;
void citire(struct pers *);
void afisare(struct pers *);
int main()
{int i,n;
```

```
dp=&p[0];
printf("\n nr. angajati:");scanf("%d",&n);
dp=(struct pers *)malloc(n*sizeof(struct pers));
if (dp==NULL)
{ printf("nu exista suficienta memorie!\n");   return 0;}
printf("Introduceti datele fiecarui angajat:\n");
for (i=0;i<n;i++)
{printf("Datele persoanei %d:\n", i+1); citire(dp+i);}
printf("Afisarea datelor introduse:\n");
for (i=0;i<n;i++)
{printf("Datele persoanei %d:\n", i+1);  afisare(dp+i);}
if (dp) free(dp);
return 0;}
void citire(struct pers *dp)
{       printf("Nume:"); scanf("%s", dp->nume);
        printf("Prenume:");  scanf("%s", dp->prenume);
        printf("Cod:"); scanf("%d", &dp->cod);
        printf("ziua nasterii:");  scanf("%d", &dp->datan.ziua);
        printf("luna nasterii:");  scanf("%d", &dp->datan.luna);
        printf("anul nasterii:");  scanf("%d", &dp->datan.an);}
void afisare(struct pers *dp)
{       printf("Nume:%s",dp->nume);
        printf("\nPrenume:%s",dp->prenume);
        printf("\nCod: %d",dp->cod);
        printf("\ndata nasterii:%d/%d/%d",dp->datan.ziua,dp->datan.luna, dp-> datan.an ); //
        printf("\nVarsta: %d ani\n",2019-dp->datan.an );}
```

Varianta in C++

```
#include <iostream>
#include <malloc.h>
using namespace std;
struct data {
        int ziua; //definitie globala de tip
        int luna;
        int an; };
struct pers {
char nume[15];
         char prenume[20];
         int cod;
         struct data datan;
         int varsta; };
struct pers p[20],*dp;
void citire(struct pers *);
void afisare(struct pers *);
int main()
{int i,n;
dp=&p[0];
cout<<"\n nr. angajati:";cin>>n;
dp=(struct pers *)malloc(n*sizeof(struct pers));
if (dp==NULL)
{ cout<<"nu exista suficienta memorie!"<<endl;   return 0;}
cout<<"Introduceti datele fiecarui angajat:"<<endl;
for (i=0;i<n;i++)
{cout<<"Datele persoanei"<< i+1<<":"; citire(dp+i);}
```

```
cout<<"Afisarea datelor introduse:"<<endl;
for (i=0;i<n;i++)
{cout<<"Datele persoanei"<<i+1<<":"<<endl;  afisare(dp+i);}
if (dp) free(dp);
return 0;}
void citire(struct pers *dp)
{       cout<<"Nume:"; cin>> dp->nume;
        cout<<"Prenume:";  cin>>dp->prenume;
        cout<<"Cod:"; cin>>dp->cod;
        cout<<"ziua nasterii:";  cin>>dp->datan.ziua;
        cout<<"luna nasterii:";  cin>>dp->datan.luna;
        cout<<"anul nasterii:";  cin>>dp->datan.an;}
void afisare(struct pers *dp)
{       cout<<"Nume:"<<dp->nume<<endl;
        cout<<"Prenume:"<<dp->prenume<<endl;
        cout<<"Cod:"<<dp->cod<<endl;
        cout<<"data nasterii:"<<dp->datan.ziua<<"/"<<dp->datan.luna<<"/"<< dp-> datan.an<<endl ; //
        cout<<"Varsta:"<<2019-dp->datan.an<<"  ani"<<endl;}
```

Rezultate:

```
 nr. angajati:2
Introduceti datele fiecarui a
Datele persoanei1:Nume:pop
Prenume:vasile
Cod:1212
ziua nasterii:02
luna nasterii:12
anul nasterii:1966
Datele persoanei2:Nume:rusu
Prenume:ana
Cod:2122
ziua nasterii:12
luna nasterii:12
anul nasterii:1971
Afisarea datelor introduse:
Datele persoanei1:
Nume:pop
Prenume:vasile
Cod:1212
data nasterii:2/12/1966
Varsta:53  ani
Datele persoanei2:
Nume:rusu
Prenume:ana
Cod:2122
data nasterii:12/12/1971
Varsta:48  ani
```

Aplicație:

Să se modifice programul astfel încât să se verifice dacă data nașterii este introdusă corect și să se ordoneze persoanele în ordinea descrescătoare a vârstei.

Ex.5. Să se scrie un program în care se declară o uniune cu 3 câmpuri (elemente) cărora li se atribuie valori. (în memorie se pastrează la un moment dat un singur element al uniunii).

Varianta in C		Varianta in C++

#include <stdio.h>
#include <string.h>
int main()
{union {char nume[10]; //declararea variabilei uniune de tipul union
char prenume[20] ;
char lit;} uniune;
//atribuirea de valori elementelor unei uniuni
printf("Modificarea elementelor uniunii\n");
printf("prin atribuirea de valori elementelor\n\n");
printf(" nume\tprenume\tlitera\n");
strcpy(uniune.nume, "Popescu"); printf("%10s", uniune.nume);
strcpy(uniune.prenume, "Maria"); printf(" %10s", uniune.prenume);
uniune.lit='T'; printf("\t%c\n",uniune.lit);
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume, uniune.lit);
strcpy(uniune.prenume, "Mia");
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume, uniune.lit);
uniune.lit='L';
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume, uniune.lit);
return 0;
}
		#include <iostream>
		#include <string.h>
		using namespace std;
		int main()
		{union {char nume[10]; //declararea variabilei uniune de tipul union
		char prenume[20] ;
		char lit;} uniune;
		//atribuirea de valori elementelor unei uniuni
		cout<<"Modificarea elementelor uniunii"<<endl;
		cout<<"prin atribuirea de valori elementelor"<<endl<<endl;
		cout<<"nume\tprenume\tlitera"<<endl;
		strcpy(uniune.nume, "Popescu ");
		cout<<uniune.nume;
		strcpy(uniune.prenume, "Maria"); cout<<uniune.prenume;
		uniune.lit='T';
		cout<<"\t"<<uniune.lit<<endl;
		cout<<uniune.nume<<" "<<uniune.prenume<<"\t"<<uniune.lit<<endl;
		strcpy(uniune.prenume, "Mia");
		cout<<uniune.nume<<" "<<uniune.prenume<<"\t"<<uniune.lit<<endl;
		uniune.lit='L';
		cout<<uniune.nume<<" "<<uniune.prenume<<"\t"<<uniune.lit<<endl;
		return 0;
		}

Rezultate:

```
Modificarea elementelor uniunii
prin atribuirea de valori elementelor

      nume      prenume litera
   Popescu      Maria   T
     Taria      Taria   T
       Mia        Mia   M
       Lia        Lia   L
```

Aplicație:

Să se modifice programul astfel încât să se introducă de la tastatură toate câmpurile uniunii.

Ex.6. În programul următor se declară un tablou unidimensional de maxim 10 uniuni, numit uniune[10], fiecare uniune conține 2 elemente care se inițializează cu valori introduse de la tastatură și ulterior se afișează. Se observă că nu se păstrează pentru fiecare element al uniunii decât ultima valoare de tip șir atribuită.

Varianta in C		Varianta in C++

#include <stdio.h>
#include <string.h>
int main()
{union { char nume[10];
char prenume[20] ;} uniune[10];
int i,n;
printf("n="); scanf("%d", &n);
for(i=0;i<n;i++)
{printf("\nNume %d:",i+1);
scanf("%s",uniune[i].nume); printf("%s",
uniune[i].nume) ;
printf("\nPrenume
%d:",i+1);scanf("%s",uniune[i].prenume);
printf("%s",uniune[i].prenume);
}
for(i=0;i<n;i++)
{printf("\nNume %d: %s",i+1, uniune[i].nume);
printf("\nPrenume %d: %s",i+1, uniune[i].prenume);
}
return 0;
}
		#include <iostream>
		#include <string.h>
		using namespace std;
		int main()
		{union { char nume[10];
		char prenume[20] ;} uniune[10];
		int i,n;
		cout<<"n="; cin>>n;
		for(i=0;i<n;i++)
		{cout<<"\nNume " <<i+1<<":";
		cin>>uniune[i].nume;
		cout<<uniune[i].nume ;
		cout<<"\nPrenume " <<i+1<<":";
		cin>>uniune[i].prenume;
		cout<<uniune[i].prenume; }
		for(i=0;i<n;i++)
		{cout<<"\nNume "
		<<i+1<<":"<<uniune[i].nume;
		cout<<"\nPrenume "
		<<i+1<<":"<<uniune[i].prenume; }
		return 0;
		}	

Rezultate:

```
n=2

Nume 1:Pop
Pop
Prenume 1:Ioan
Ioan
Nume 2:Rusu
Rusu
Prenume 2:Alina
Alina
Nume 1: Ioan
Prenume 1: Ioan
Nume 2: Alina
Prenume 2: Alina
```

Aplicație:

Să se modifice programul astfel încât să se afișeze și inițiala numelui introdus pentru fiecare uniune

Ex.7. În programul de mai jos se declară 4 pointeri la șiruri de caractere, *pwest, *pnorth,*peast,*psouth ***și o variabilă numită*** direction ***la tipul enumerare numit*** location***. Afișarea valorilor șirurilor se realizează utilizând pointerii declarați anterior și valorile enumerării.***

Varianta in C

```
#include <stdio.h>
int main()
{char *pwest = "Vest",*pnorth = "Nord", *peast="Est", *psouth = "Sud";
//declararea tipului enumerare
enum location { Est=1, Vest=2, Sud=3, Nord=4};
//declararea variabilei direction de tip enumerare
enum location direction;
printf("Introducesti directia: Est=1, Vest=2, Nord=3, Sud=4\n");
scanf("%d", &direction);
if( direction == 1 )  printf("Directia indicata este %s\n", peast);
if( direction == 2 ) printf("Directia indicata este %s\n", pwest);
if( direction == 3 )  printf("Directia indicata este %s\n", pnorth);
if( direction == 4 ) printf("Directia indicata este %s\n", psouth);
return 0;}
```

Rezultate:

```
Introducesti directia: Est=1, Vest=2, Nord=3, Sud=4
2
Directia indicata este Vest
```

Aplicație:

Să se scrie o altă variantă de program utilizând instrucțiunea switch

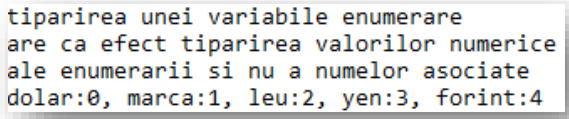
Ex.8 Programul este un exemplu de definire a unui tip de enumerare numit monede ***și a unei variabile de acest tip numite*** bani ***si ilustrează accesul la elementele variabilei și modul de tipărire a acestora.***

Varianta in C		Varianta in C++

#include <stdio.h>
int main()
{//declarare tip enumerare
enum monede {dolar,marca,leu,yen,forint } ;
//declarare variabila enumerare
enum monede bani;
bani=leu;
if (bani==forint) printf("Moneda este un forint\n");
printf("tiparirea unei variabile enumerare\n");
printf("are ca efect tiparirea valorilor numerice\nale enumerarii ");
printf("si nu a numelor asociate \n");
printf("dolar:%d, marca:%d, leu:%d, yen:%d, forint:%d\n",
dolar,marca,leu,yen,forint); //atentie linie continuata!
return 0;
}
		#include <iostream>
		using namespace std;
		int main()
		{//declarare tip enumerare
		enum monede {dolar,marca,leu,yen,forint } ;
		//declarare variabila enumerare
		enum monede bani;
		bani=leu;
		if (bani==forint) cout<<"Moneda este un forint"<<endl;
		cout<<"tiparirea unei variabile enumerare"<<endl;
		cout<<"are ca efect tiparirea valorilor numerice\nale enumerarii ";
		cout<<"si nu a numelor asociate "<<endl;
		cout<<"dolar:"<<dolar<<",
		marca:"<<marca<<", leu:"<<leu<<",
		yen:"<<yen<<", forint:"<<forint<<endl;
		//atentie linie continuata!
		return 0;}

Rezultate:

Aplicație:



Să se modifice programul astfel încât declarația variabilei să fie următoarea:
enum monede
{dolar,marca,leu=100,yen,forint } ;
Ce se va afișa în acest caz?

Ex.9. Programul implementeaza o bază de date cu calculatoare, fiecare articol cu câmpurile: denumire, garanție, și preț. Se citesc articolele de la tastatură și se afișeaza pe monitor.

Varianta in C

```
#include <stdio.h>
#include <string.h>
int main()
{int i,n;
typedef struct { char den[10];
                 int gar;   //garantie
                 double pret;
                 }calculator;
calculator  j[10];
printf("Nr. de calculatoare:"); scanf("%d",&n);
printf("Introducerea calculatoarelor:");
for (i=0;i<n;i++)
{printf("\nCalculatorul nr. %d\n", i+1);
printf("Denumire:"); scanf("%s", j[i].den);
printf("Garantie:"); scanf("%d", &j[i].gar);
printf("Pret(lei):"); scanf("%lf", &j[i].pret);}
printf("\nAfisarea produselor introduse:\n");
for (i=0;i<n;i++)
{printf("%10s,%d,%5.2lf lei", j[i].den, j[i].gar, j[i].pret);}
printf("\n");
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{int i,n;
typedef struct { char den[10];
                 int gar;   //garantie
                 double pret;
                 }calculator;
calculator  j[10];
cout<<"Nr. de calculatoare:"; cin>>n;
cout<<"Introducerea calculatoarelor:";
for (i=0;i<n;i++)
{    cout<<"Calculatorul nr. "<<i+1<<endl;
cout<<"Denumire:"; cin>> j[i].den;
cout<<"Garantie:"; cin>>j[i].gar;
cout<<"Pret(lei):"; cin>>j[i].pret;}
cout<<endl<<"Afisarea produselor introduse:"<<endl;
for (i=0;i<n;i++)
```

```
{cout<<j[i].den<<", "<<j[i].gar<<","<<j[i].pret<<" lei";
cout<<endl;}
for (i=0;i<n;i++)
{cout<<j[i].den<<", "<<j[i].gar<<","<<j[i].pret<<" lei";
cout<<endl;}
return 0;} #include <iostream>
#include <string.h>
using namespace std;
int main()
{int i,n;
typedef struct { char den[10];
                 int gar;   //garantie
                 double pret;
                 }calculator;
calculator  j[10];
cout<<"Nr. de calculatoare:"; cin>>n;
cout<<"Introducerea calculatoarelor:";
for (i=0;i<n;i++)
{    cout<<"Calculatorul nr. "<<i+1<<endl;
cout<<"Denumire:"; cin>> j[i].den;
cout<<"Garantie:"; cin>>j[i].gar;
cout<<"Pret(lei):"; cin>>j[i].pret;}
cout<<endl<<"Afisarea produselor introduse:"<<endl;
for (i=0;i<n;i++)
{cout<<j[i].den<<", "<<j[i].gar<<","<<j[i].pret<<" lei";
cout<<endl;}
for (i=0;i<n;i++)
{cout<<j[i].den<<", "<<j[i].gar<<","<<j[i].pret<<" lei";
cout<<endl;}
return 0;}
```

Rezultate:

```
Nr. de calculatoare:2
Introducerea calculatoarelor:Calculatorul nr. 1
Denumire:HP
Garantie:2
Pret(lei):3000
Calculatorul nr. 2
Denumire:COMPAQ
Garantie:5
Pret(lei):4000

Afisarea produselor introduse:
HP, 2,3000 lei
COMPAQ, 5,4000 lei
```

Aplicație:

Să se modifice programul astfel incat sa se afiseze doar articolele care au prețul in intervalul (1000,2500)

PROBLEME PROPUSE

1. ***Să se scrie un program care realizează citirea a 5 date calendaristice şi memorarea acestora într-un tablou unidimensional de structuri. Se vor afişa datele pentru care anul este mai mare decât 2004.***
2. ***Să se scrie un program care citeşte numele, prenumele,data naşterii, codul personal pentru n persoane, calculează vârsta fiecarei persoane şi apoi afişează toate datele, utilizând alocarea dinamică a memoriei.***
3. ***Să se scrie un program în care se defineşte o variabilă tablou de structuri numită angajați de tip structură cu câmpurile: nume, adresa, cod numeric, salar net şi să se inițializeze tabloul cu n articole. Să se afişeze numai angajaţii care au salarul cuprins între 400 şi 1.000 RON.***
4. ***Să se scrie un program în care se defineşte o structură de tip catalog de carţi cu următoarele câmpuri: titlu, autor, editura, anul apariţiei. Să se definească o variabilă tablou de structuri de tipul catalog în care iniţializarea datelor şi afişarea lor să se realizeze atât prin intermediul câmpurilor cât şi prin intermediul unui pointer la această variabilă.***
5. ***Să se scrie un program în care se defineşte o variabilă tablou de structuri (magazin de tehnică de calcul) de tipul structură cu câmpurile:***
   - ***Denumire (alfanumeric, max.30 caractere). Ex. : PC Compaq P910***
   - ***Tip (alfabetic, max.10 caractere). Ex. calculatoare***
   - ***Caracterisitici (alfabetic, max.30 caractere). Ex. 800MHz, 10 GB HDD***
   - ***Garanție (numeric intreg, max. 2 cifre) . Ex. 3***
   - ***Preț în euro (numeric real , max. 5 cifre). Ex. 950***

   Se cere să se citească de la tastatură n articole cu formatul de structură de mai sus, să se calculeze preţul echivalent in RON și să se afișeze articolele, utilizând formatul de mai jos: Denumire Pret (RON) PC Compaq P910 2 850 RON .

6. ***Se consideră o bază de date, de tipul unui magazin de echipamente electronice, în care fiecare echipament reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Denumire (alfanumeric,max.30 caractere).Ex.:Video Recorder***
   - ***Cod (alfanumeric, max 6 caractere). Ex. A254G9***
   - ***Garanție (numeric intreg, max. 2 cifre) . Ex. 3***
   - ***Culoare (enumerare de maxim 5 culori) . Ex. alb, gri, negru, argintiu***
   - ***Preț (numeric real , max. 10 cifre). Ex. 450***

   Să se scrie programul în care să se citeasca n articole (n introdus de la tastatură) de tipul specificat mai sus, iar afişarea articolelor să se realizeze utilizând numai câmpul denumire şi preţul calculat în EURO la cursul oficial BNR din ziua respectivă (curs citit de la tastatură).