13. Structuri.

13.1. Definirea tipurilor structurilor şi declararea variabilelor structuri.

Structura este o colecție de variabile (care vor fi numite în continuare *membri*, *câmpuri* sau *selectori*) grupate sub un singur nume. Structura este eterogenă, adică poate cuprinde membri de tipuri diferite. Câmpurile structurii se declară prin nume şi tip (ca variabilele). Declararea unei structuri se face prin:

```
struct nume_structură { declarații_câmpuri; };
```

După acolada închisă din declarația câmpurilor se pune întotdeauna ; .
Prin declararea unei structuri nu se rezervă memorie ci se defineşte un tip. Exemple*:*

```
struct complex {double re; double im;};
struct data {int an; int luna; int zi;};
struct persoana { char* nume;
                  char* strada;
                  int numar;
                  int varsta;};
struct student { struct persoana stud;
                  char* grupa;
                  int an;};
```

În cazul în care câmpurile unei structuri sunt pointeri, este necesar să alocăm dinamic memorie pentru datele referite de acei pointeri.
Alocarea de memorie se face în momentul definirii unor variabile structuri, folosind numele dat structurii:

```
struct nume_structură listă_variabile;
```

Exemple:

```
struct complex z1, z2;
struct data d;
struct persoana p;
struct student s;
```

Declararea structurii poate fi combinată cu definirea variabilelor de tip structură:

```
struct nume_structură {declaratii campuri;} lista_variabile;
```

Exemple:

```
struct complex { double re; double im;} z1, z2;
struct data {int an; int luna; int zi;} d;
```

Numele structurii poate lipsi, practică pe care nu o recomandăm:

```
struct { declaratii_campuri; } lista_variabile;
```

Exemplu:

```
struct {double re; double im;} z1, z2;
```

În C, folosirea declarației **typedef** ne permite o utilizare mai comodă a structurilor:

```
typedef tip nume;
```

```
struct complex {double re; double im;};
typedef struct complex COMPLEX;
COMPLEX z1, z2;
```

O formă combinată a declarării structurii cu **typedef** este următoarea:

```
typedef struct {double re; double im;} COMPLEX;
COMPLEX z1,z2;
```

13.2. Inițializarea structurilor.

Inițializarea variabileleor structuri se poate face la definirea lor, valorile inițiale fiind cuprinse între acolade.
Exemple:

```
COMPLEX z={1.5,-2.0};
struct persoana p={"Popescu Ion","Popa Nan",72,53};
```

Dacă structurile au câmpuri tablouri sau alte structuri, la inițializare se pot folosi mai multe niveluri de acolade.
Exemple:

```
typedef struct stud {char *nume;
                     int note[5];
                     data nast;} STUDENT;
STUDENT s={"Popa Vasile",{9,7,10,8,6},{1980,9,15}};
```

13.3. Accesul la câmpurile structurilor.

Pentru a obține acces la membrii unei structuri vom folosi operatorul punct:

```
variabilă_structură.nume_membru
```

Exemple:

```
COMPLEX z;
struct persoana p;
struct student s;
z.re=1.5;
z.im=-2.0;
p.nume = strdup("Popa Vasile");
s.stud.nume = strdup("Vasilescu Paul");
```

13.4. Pointeri la structuri.

Considerăm definițiile de pointeri la structuri:

```
COMPLEX z, *pz=&z;
struct persoana p, *pp=&p;
```

Adresarea la membrii structurilor prin intermediul pointerilor se face dereferențiind pointerii:

```
(*pz).re=1.5;
(*pp).nume=strdup("Popescu Ion");
```

sau folosind un nou operator de selecție indirectă −>:

```
pz->re=1.5;
pp->nume=strdup("Popescu Ion");
```

13.5. Atribuiri de structuri.

Două variabile având acelaşi tip structură pot apare ca termeni în atribuiri:

```
COMPLEX z1,z2,*pz;
z1=z2;
pz=&z2;
z1=*pz;
```

Am văzut că nu putem copia tablouri, şi în particular, şiruri de caractere prin atribuire. Totuşi, este posibil să realizăm atribuirea pe şiruri de caractere, declarându-le ca structuri:

```
typedef struct {char x[100];} STRING;
STRING s1,s2;
s1=s2; // !!
```

Atribuirea unei structuri reprezintă o copiere bit cu bit.

13.6. Structuri şi funcţii.

O funcţie poate întoarce ca rezultat:

- o structură

```
// functie ce intoarce o structura
COMPLEX cmplx(double x, double y)
{ COMPLEX z;
  z.re=x;
  z.im=y;
  return z;
}
```

- un pointer la o structură

```
// functie ce intoarce un pointer la o structura
COMPLEX *cmplx(double x, double y)
{ COMPLEX z, *pz = &z;  //functioneaza corect
  pz->re=x;
  pz->im=y;
  return pz;
}
```

- un membru al unei structuri

```
// functie ce intoarce un membru al structurii
double real(COMPLEX z)
{ return z.re;
}
```

Parametrii unei funcţii pot fi:

- structuri

```
// functie ce intoarce o structura
```

```
COMPLEX conjg(COMPLEX z)
{ COMPLEX w;
  w.re=z.re;
  w.im=-z.im;
  return w;
}
```

- pointeri la structuri

```
void conjg(COMPLEX *pz){
 py->im=-pz->im;
}
```

Funcția următoare inițializează, prin citire, o structură de tip **persoană** (tip definit mai sus). Funcția îşi poate realiza efectul:

- prin modificarea unei structuri adresate printr-un pointer

```
void date_pers(struct persoana *p)
{ char buf[80];
  p->nume=strdup(gets(buf));
  p->strada=strdup(gets(buf));
  scanf("%d",&(p->numar));
  scanf("%d",&(p->varsta));
}
```

- prin rezultatul structură întors de funcție:

```
struct persoana date_pers(void)
{ struct persoana p;
  char buf[80];
  p.nume=strdup(gets(buf));
  p.strada=strdup(gets(buf));
  scanf("%d",&(p.numar));
  scanf("%d",&(p.varsta));
}
```

Exemplu 32: ***Să se calculeze numărul de zile dintre două date calendaristice***.

Vom incrementa în mod repetat cea mai mică dintre date, până când devine egală cu cea de-a doua.

```
#include <stdio.h>
typedef struct {int an,luna,zi;} DATA;
int citdata(DATA *);
int nrzile(DATA, DATA);
int dateegale(DATA, DATA);
int precede(DATA, DATA);
void incrdata(DATA *);
int bisect(DATA);
int main(){
  int n;
  DATA d1, d2;
  citdata(&d1);
```

```
  citdata(&d2);
  if ((n=nrzile(d1, d2))>0)
    printf("intre cele doua date sunt %d zile\n",n);
  else
    printf("prima dintre date este dupa cealalta\n");
  return 0;
}

int citdata(DATA *d){
  if(scanf("%d %d %d",&d->an,&d->luna,&d->zi)==3 &&
           d->an > 0 &&
           d->luna > 0 && d->luna < 13 &&
           d->zi > 0 && d-> zi < 32)
     return 1;
  else
    { printf("Data incorecta\n");
      return 0;
    }
}

int dateegale(DATA d1, DATA d2){
 return(d1.an==d2.an && d1.luna==d2.luna && d1.zi==d2.zi);
}

int precede(DATA d1, DATA d2){
 return(d1.an < d2.an ||
        d1.an==d2.an && d1.luna < d2.luna ||
        d1.an==d2.an && d1.luna==d2.luna && d1.zi < d2.zi);
}

int bisect(DATA d){ return (d.an%4==0 && d.an%100!=0 || d.an
%400==0);}

int nrzile(DATA d1, DATA d2){
 int i=0;
  if(precede(d1,d2)) {
    while(!dateegale(d1,d2)) {
      incrdata(&d1);
      i++;
    }
    return i;
  }
  else
    return 0;
}

void incrdata(DATA *d){
  static
   ultima[2][13]={{0,31,28,31,30,31,30,31,31,30,31,30,31},
```

```
                    {0,31,29,31,30,31,30,31,31,30,31,30,31}};
  if(d->zi < ultima[bisect(*d)][d->luna])
    d->zi++;
  else
    { d->zi=1;
      if(d->luna < 12)
        d->luna++;
      else
        { d->luna=1;
          d->an++;
        }
    }
}
```

13.7. Uniuni.

Folosirea uniunilor permite suprapunerea în acelaşi spaţiu de memorie a mai multor variabile. Conținutul zonei de memorie poate fi interpretat în mod diferit.
Definirea unei uniuni se face ca şi a unei structuri. Uniunea poate fi considerată o structură, în care toți membrii au deplasamentul 0 față de adresa de început. Uniunea este suficient de mare pentru a conține cel mai voluminos membru, respectând restricțiile de aliniere.
De exemplu pentru a interpreta conținutul unei zone de memorie ca un întreg, un real sau un şir de caractere, le vom suprapune într-o uniune:

```
union supra {int i; double d; char* c;};
union supra u;
```

sau

```
typedef union {int i; double d; char* c;} SUPRA;
SUPRA u;
```

Accesul se face ca în cazul structurilor:

```
u.i;    // pentru intreg (primii 2 octeti)
u.d;    // pentru real (primii 8 octeti
u.c[0]; // pentru primul octet caracter
```

Exemplul 33: ***Definiți o funcție care afişează reprezentarea internă a unui număr real***.

```
void hexdouble(double x)
{ union {double d; char c;} u;
  unsigned n=sizeof(double);
  char *p = &(u.c);
  u.d=x;
  while(n--)
    printf("%02x ",*p++);
}
```

13.8. Probleme propuse.

1.Scrieți o funcție având ca parametri două date calendaristice (precizate prin an, lună şi zi), care stabileşte una din situațiile:

- Prima dată o precede pe cea de.a doua
- Cele două date sunt egale
- A doua dată o precede pe prima

Funcția va întoarce una din valorile -1, 0, 1.

2. La o disciplină cu notare pe parcurs, fiecărui student i s-au acordat 3 note la trei lucrări de control şi un calificativ pentru activitatea la seminar (insuficient, suficient, bine, foarte bine). Pe baza acestor informații se acordă o notă finală în felul următor: se face media celor 3 note la care se adaugă 0 pentru insuficient, 0.25 pentru suficient, 0.5 pentru bine şi 0.75 pentru foarte bine, şi apoi rezultatul se trunchiază.

- Să se definească tipul situatie ca o structură având ca membri câmpurile nume – un şir de 20 de caractere
  note – un tablou de 3 întregi
  calificativ – un caracter
- Să se definească o funcție având ca parametru o situație, care calculează nota finală.
- Să se scrie un program care citeşte situațiile a n studenți şi afişează lista studenților promovați (având nota finală ≥ 5).

3. Definiți tipul mulțime de reali ca o structură cu următoarele câmpuri:
- numărul de elemente (o valoare întreagă)
- valorile elementelor (un tablou de reali, având cel mult 100 de elemente)

Se vor defini funcții pentru:

- a stabili dacă o valoare dată x aparține sau nu unei mulțimi date M
- creerea mulțimii diferență a două mulțimi.
- Calculul valorii unui polinom (definit printr-o structură identică cu a mulțimii de reali) într-un punct dat x
- Calculul polinomului derivat. Această funcție are doi parametri structuri: polinomul dat şi polinomul derivat şi nu întoarce rezultat.

Funcția **`main()`** :

- citeşte un polinom dat prin gradul **`n`** şi cei **`n+1`** coeficienți
- citeşte cele r posibile rădăcini ale polinomului
- stabileşte pentru fiecare rădăcină multiplicitatea ei.

Indicație: Componentele **`x[k]`** pentru care **`P(x[k]) = 0`** sunt rădăcini cu multiplicitate cel puțin 1, cele pentru care **`P'(x[k]) = 0`** sunt rădăcini cu multiplicitate cel puțin 2, ş.a.m.d. Rădăcinile simple se obțin făcând diferența dintre mulțimea rădăcinilor cu multiplicitate cel puțin 1 şi cele cu multiplicitate cel puțin 2, etc.

4. a) Definiți structurile:

- **`complex`** – având doi membri reali (partea reală şi partea imaginară;

- **polinom** – cu membrii: **grad** – un întreg, **coeficienți** – un tablou de valori complexe.

b) Definiți funcții pentru: adunarea şi înmulțirea a două numere complexe şi pentru calculul valorii unui polinom într-un punct dat.

c) Scrieți o funcție **main()** care:

- citeşte un polinom, de grad cel mult 20
- citeşte un număr complex **z**
- calculează şi afişează valoarea polinomului în punctul **z**.

5. Un polinom este precizat printr-o structură având câmpurile: **grad** – un întreg şi **coeficienți** – un tablou de reali. Vom considera gradul cel mult 20.

a) Definiți o funcție care calculează valoarea unui polinom într-un punct dat. Funcția are ca parametri un polinom **P** şi un real x şi întoarce ca rezultat un real – valoarea **P(x)**.

b) Definiți o funcție care derivează un polinom. Funcția are ca parametri două structuri: polinomul dat şi polinomul derivat şi nu întoarce nimic.

c) Scrieți o funcție **main()** care:

- Citeşte un polinom **P**
- Citeşte o valoare reală **x**
- Stabileşte şi afişează multiplicitatea rădăcinii **x** a polinomului **P**.

6. a) Să se definească tipurile **punct, dreptunghi, cerc** si **nor** de puncte ca tipuri înregistrare. Tipul **punct** va avea drept câmpuri abscisa **x** şi ordonata **y** a punctului, tipul **cerc** va avea drept câmpuri **centrul** - un punct şi **raza** -o valoare reala, tipul **dreptunghi** -colturile stânga-jos şi dreapta-sus a două vârfuri opuse ale dreptunghiului,care sunt puncte, iar tipul **nor de puncte** - numărul de puncte din nor şi coordonatele lor.

b) Să se definească funcții având ca parametri un **punct** şi un **dreptunghi** (respectiv un **cerc**), care stabilesc dacă punctul este sau nu interior dreptunghiului (cercului).

Să se scrie un program care citeşte:un nor de puncte,un dreptunghi şi un cerc şi care stabileşte câte puncte din nor sunt interioare dreptunghiului şi câte cercului şi afişează aceste informații.

7. Pentru aprovizionarea unui magazin se lansează **n** comenzi (**n <= 100**). O comandă este precizată prin două elemente: **tipul** produsului comandat (un tablou de 8 caractere) şi **cantitatea** comandată (o valoare întreagă). Un produs poate fi comandat de mai multe ori. Să se centralizeze comenzile pe produse, astfel încât comenzile centralizate să se refere la produse diferite.

8. .Să se rezolve ecuația $a.x^3 + b.x^2 + c.x + d = 0$, cu $a, b, c, d \in R$, $a \neq 0$, folosind formulele lui Cardan.

9. O matrice rară, adică o matrice având majoritatea elementelor nule se memorează economic într-o înregistrare conținând: numărul de linii, numărul de coloane, numărul de elemente nenule, precum şi doi vectori, unul cu elementele nenule din matrice ,iar celă lalt cu pozițiile lor, facând vectorizarea matricii pe linii.

Să se definească funcţii pentru adunarea şi înmulţirea a două matrici rare, precum şi pentru creerea structurii matrice rară şi afişarea acesteia ca o matrice.
Se va scrie un program care citeşte şi afişează două matrici rare şi care apoi le adună şi le înmulteşte, afişând de fiecare dată rezultatele.

10. a) Să se definească tipul punct ca tip înregistrare.
b) Să se definească o funcţie având ca parametri trei puncte care stabileşte dacă acestea sunt sau nu coliniare.
c) Să se scrie un program care citeşte un întreg **n** (**n <= 50**) şi n puncte şi afişează numerele tripletelor de puncte coliniare.

11. Un bilet pronosport conţine numele jucatorului şi 13 caractere **1, 2, x** constituind reprezentarea codificată a rezultatelor unor meciuri de fotbal.
a) Să se descrie structura unui bilet pronosport folosind tipul înregistrare.
b) Să se definească o funcţie având ca parametri doi vectori de acelaşi tip, funcţie care stabileste numărul de componente egale din cei doi vectori.

- c) Să se scrie un program care primeşte ca date n bilete pronosport, precum şi rezultatele a 13 meciuri jucate (codificate 1, 2, x) şi afiseaza:

lista jucatorilor, pentru fiecare indicându- se numărul de pronosticuri exacte

- lista jucatorilor, ordonată după numărul de pronosticuri exacte indicate.

Numele jucatorilor se citesc începând din coloana 1, iar pronosticurile din 21.

12. Un experiment fizic este precizat prin: numarul de determinări şi valorile măsurate.
a) Să se descrie structura experiment folosind tipul înregistrare.
b) Să se definească o funcţie având ca parametru un experiment, care calculează media aritmetică a masuratorilor.
c)Să se scrie un program care citeste numărul de determinări şi valorile lor şi creează cu acestea o înregistrare şi calculează folosind funcţia de mai sus abaterea standard:

$$s = \sqrt{\frac{\sum_{i=0}^{n-1} x_i^2 - n\sum_{i=0}^{n-1}\left(\frac{x_i}{n}\right)^2}{n-1}}$$

13. a) O dată calendaristică este exprimată prin trei valori întregi: anul, luna şi ziua; o persoană este precizată prin: nume şi prenume (maxim 30 de caractere) şi data naşterii - o dată calendaristică. Să se descrie tipurile dată şi persoană ca tipuri înregistrare.
b) Să se definească o funcţie având ca parametru o persoană, funcţie care calculează vârsta persoanei în ani împliniţi.
c) Să se scrie un program care citeşte o listă de n persoane (**n** este citit înaintea listei) şi datele lor de naştere şi folosind funcţia definită mai sus afişează lista persoanelor majore.

14. a) Să se descrie tipurile punct şi dreaptă ca tipuri înregistrare.
b) Să se definească o funcţie având ca parametri două drepte, un punct şi o variabilă booleană, funcţie care stabileşte dacă cele două drepte se intersectează, caz în care calculează coordonatele

punctului de intersecție, sau sunt paralele; parametrul boolean separă situația drepte paralele/ concurente.

c) Să se scrie un program care citeşte **n** drepte (**n <= 100**) şi afişează perechile de drepte paralele, iar pentru fiecare pereche de drepte neparalele - coordonatele punctului de intersecție. De exemplu:

Drepte paralele:

```
1 – 3
2 – 4
```

Drepte concurente:

```
1 – 2     (8.0,  7.0)
1 – 4     (17.0, 5.0)
2 – 3     (4,0,  4.0)
3 – 4     (13.0, 2.0)
```

15. a) Să se definească tipurile punct şi triunghi ca tipuri înregistrare.

b) Să se definească o funcție având ca parametru un triunghi funcție care calculează aria acestuia.

c)Să se definească o funcție având ca parametri un punct şi un triunghi, funcție care stabileşte dacă punctul este interior sau exterior triunghiului. (dacă punctul **M** este interior triunghiului **ABC** atunci **aria(ABC)=aria(MAB)+aria(MBC)+aria(MCA)).**

d) Să se scrie un program care citeşte 4 puncte şi determină folosind funcția de la punctul c) dacă acestea pot forma un patrulater convex.