TIPURI COMPLEXE DE DATE – STRUCTURI

O structură este o colecție de una sau mai multe variabile (de același tip sau de tipuri diferite) grupate sub un singur nume. Variabilele care fac parte din structură se numesc membrii structurii respective. Membrii unei structuri pot fi tipuri de date simple (int, float, etc), tipuri de date agregate (tablouri, alte tipuri de structuri) sau pointeri (la diverse tipuri de date, inclusiv pointer la același tip de structură – structuri autoreferite).

1. Declarația și definiția unei structuri

1.1. Declarația unei structuri

Forma generală a declarației unei structuri este:

```
struct NUME_STRUCTURĂ {
     declarații de date;
};
```

și cuprinde:

- Cuvântul cheie **struct**
- Numele modelului (prototipului) structurii **NUME_STRUCTURĂ**
- Descrierea membrilor structurii respective. Forma generală a membrilor unei structuri este
  - `tip nume;`

sau

  - `tip nume[dim1][dim2]…[dimn];`

Prin această descriere se indică "prototipul" (modelul) structurii respective, nu se rezervă zonă de memorie și această declarație poate apare într-un fișier header.

O structură se poate declara și fără indicarea componenței sale, adică putem scrie:

```
struct NUME_STRUCTURĂ;
```

urmând ca descrierea să se facă ulterior.

Declarația unei structuri va apare în fișierul header înainte oricăror referiri la tipul de structură respectiv (de exemplu, înaintea specificării prototipurilor funcțiilor care folosesc ca parametru sau ca valoare de retur o structură de tipul specificat).

1.2. Definiția unei structuri

Are rolul de a rezerva zonă de memorie pentru variabilele de tip structură forma generală este:

În cazul în care structura a fost declarată anterior (în fișierul header), definirea unor variabile de tip structură se face în forma:

```
struct NUME_STRUCTURĂ v1, v2[10], …, vn;
```

unde v1, v2[10], …, vn sunt variabile de tip structură al cărui model este NUME_STRUCTURĂ. v2[10] este un tablou unidimensional în care fiecare element este o structură cu modelul NUME_STRUCTURĂ.

Definiția unei variabile de tip structură nu poate apare într-un fișier header.

2. Operații cu structuri

2.1. Inițializarea

Forma generală pentru inițializarea unei structuri este:

```
struct NUME_STRUCTURĂ variabila_structura = {val1, …, valn};
```

val_1, …, val_n reprezintă valorile de inițializare ale membrilor structurii respective și trebuie să coincidă cu tipul membrului structurii la care se referă.

Exemplu:

Fiind dată o declarație de structură care descrie o dată calendaristică de forma:

```
struct DATA {
     int zi;
     char luna[15];
     int an;
};
```

Definirea cu inițializare a unei variabile ziDeNastere de tip DATA se face prin:

```
struct DATA ziDeNastere = {11, "iunie", 1970};
```

2.2. Accesarea membrilor

Un membru al unei structuri se accesează folosind operatorul de selecție **.** (punct).

Forma generală este:

```
variabila_structura.membru_structura
```

Luând exemplul de mai sus, afișarea conținutului variabilei de tip structură **ziDeNastere** se face astfel:

```
printf("%d %s %d\n", ziDeNastere.zi, ziDeNastere.luna, ziDeNastere.an);
```

2.3. Copierea unei structuri

Este permisă copierea unei structuri în altă structură fie membru cu membru (prin atribuiri aplicate unor variabile simple), fie luând structura ca un întreg.

Exemplu:

```
struct DATA zi1 = {25, "martie", 2004};
struct DATA zi2, zi3;
zi2 = zi1;
zi3.zi = zi1.zi;
```

după secvența de cod de mai sus, membrii structurii **zi2** vor avea aceleași valori ca și membrii structurii **zi1**, iar membrul **zi** al structurii **zi3** va avea aceeași valoare cu membrul **zi** al structurii **zi1**, ceilalți doi membri rămânând cu valori nedeterminate.

2.4. Structuri și funcții

O structură poate fi folosită ca parametru de intrare pentru o funcție și o funcție poate returna o structură. Modificările făcute asupra valorii membrilor unei structuri într-o funcție nu se păstrează și după ieșirea din funcție. O structură se transmite către o funcție prin valoare, prin intermediul stivei.

Exemplu:

Fiind declarată structura următoare asociată unui punct din plan

```
struct  _PUNCT {
     double x, y;
};
typedef struct _PUNCT PUNCT;
```

a) să se scrie o funcție care returnează distanța dintre două puncte;
b) să se scrie o funcție care returnează punctul simetric față de origine al unui punct dat.

a)
```
double distanta(PUNCT a, PUNCT b)
{
     double d;
     d = sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
     return d;
}
```

b)
```
PUNCT simetric(PUNCT a)
{
     PUNCT b;
     b.x = -a.x;
     b.y = -a.y;
     return b;
}
```

Considerând următoarele definiții:

```
double d;
PUNCT A, B, X, Y;
```

și că variabilele structură de tip PUNCT A, B, Y sunt inițializate pe parcursul programului, apelul celor două funcții se face astfel:

```
/*
 *   Calculează distanța dintre punctele ale căror coordonate
 *        sunt în structurile A și B de tip PUNCT
 */

    d = distanta(A, B);

/*
 *   Membrii structurii X vor avea ca valori
 *        valorile simetrice față de origine ale membrilor structurii Y.
 */

    X = simetric(Y);
```

Pentru fiecare problemă se va construi un proiect cu fișierul header corespunzător.

TEMA 1

Problema nr. 1.1

a) Să se declare tipul de dată structură având numele modelului structurii **_FRACTIE** asociat unei fracții care are ca membri două numere întregi corespunzătoare numărătorului și numitorului unei fracții. Să se construiască apoi un sinonim pentru această structură, sinonim având numele **FRACTIE**.

b) Să se scrie funcții pentru determinarea inversei unei fracții și calculul sumei, diferenței și produsului a două fracții. Rezultatul va fi sub forma unei fracții ireductibile (care nu se poate simplifica). Pentru aceasta se va scrie o funcții care determină cel mai mare divizor comun conform algoritmului lui Euclid descris în Laboratorul nr. 2.

c) Să se scrie un program care citește dintr-un fișier numărătorul și numitorul a două fracții **p** și **q** și calculează și afișează forma ireductibilă a fracțiilor p+q, p-q, p*q și p/q în funcție de o opțiune dată de utilizator.

Problema nr. 1.2

a) Să se definească tipul **TIMP** ca o structură care conține ca membri trei întregi reprezentând ora (cu valori între 0 și 24), minutul (cu valori între 0 și 60) și secunda (cu valori între 0 și 60).

b) Să se scrie o funcție care primind ca parametri de intrare două date de tipul **TIMP** calculează suma lor într-o dată de tipul **TIMP**.

c) Să se scrie o funcție care primind ca parametri de intrare două date de tipul **TIMP** calculează diferența lor într-o dată de tipul **TIMP**.

d) Să se scrie un program care rezolvă următoare problemă: un angajat al unei firme ajunge la lucru la ora **t1** și lucrează până la ora **t2**. Pleacă apoi la masă și revine la ora **t3** și mai lucrează până la ora **t4**. Știind că momentele t1, t2, t3, t4 sunt memorate ca date de tipul **TIMP**, să se calculeze și să se afișeze (sub forma hh:mm:ss) timpul lucrat de angajat în ziua respectivă.

Problema nr. 1.3

a) Să se definească tipurile **punct**, **cerc** și **dreptunghi** ca structuri. Tipul **punct** va avea ca membri două numere reale reprezentând abscisa **x**, respectiv ordonata **y** a punctului căreia îi este asociată structura. Tipul **cerc** va avea ca membri un punct (centrul cercului) și un număr real (raza cercului), iar tipul **dreptunghi** va avea ca membri două puncte corespunzătoare colțului din stânga sus și colțului din dreapta jos.

b) Să se definească funcții având ca parametri un punct și un dreptunghi (respectiv un cerc) care stabilesc dacă punctul se găsește în interiorul dreptunghiului (respectiv cercului). Funcțiile vor returna 1 dacă punctul este interior și 0 în caz contrar.

c) Să se scrie un program care citește informații referitoare la un dreptunghi și la un cerc și stabilește dacă un punct ale căror coordonate sunt citite de la tastatură este interior sau nu acestor figuri geometrice. Programul va citi coordonatele unui punct, afișează rezultatul și apoi printr-un dialog cu utilizatorul stabilește dacă citește coordonatele unui alt punct sau se termină.

TEMA 2

Problema nr. 2.1

Se consideră următoarea structură asociată unui punct din spațiul bidimensional:

```
struct _PUNCT {
     double x;      // abscisa
     double y;      // ordonata
};
```

și declarația de tip

```
typedef struct _PUNCT PUNCT;
```

Folosind aceste două declarații să se rezolve următoarea problemă:

Se citește un număr întreg **n** și apoi coordonatele a **n** puncte din spațiul bidimensional (fiecare linie conține coordonatele unui punct separate printr-un spațiu).

Exemplu:

Avem n = 3 și coordonatele

1 1

-2 4

-4 -3

Să se calculeze aria și perimetrul poligonului determinat de cele **n** puncte, presupunând că poligonul este convex. Valorile pentru **arie** și **perimetru** se vor afișa după afișarea coordonatelor vârfurilor poligonului cu **3 zecimale**.

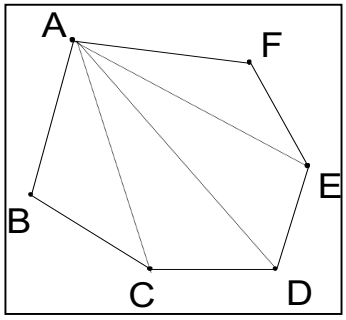
Se vor scrie și folosi funcții pentru:

- Calculul distanței dintre două puncte;
- Calculul ariei unui triunghi folosind formula lui Heron (dată în primul curs);
- Calculul semiperimetrului unui triunghi;
- Calculul ariei totale;
- Calculul perimetrului poligonului
- Citirea unui vector de structuri dintr-un fișier.
- Afișarea coordonatelor vârfurilor poligonului știind că primul vârf este notat cu **A**, iar coordonatele unui vârf se indică între paranteze după "numele" vârfului.

  Exemplu:

  ```
  A(1.00, 1.00) B(2.75, 3.14)
  ```

Pentru afișarea coordonatelor de se vor folosi două zecimale și se vor afișa câte 5 puncte pe o linie, puncte separate printr-un TAB.



Perimetrul = $\sum$ lungimilor laturilor AB, BC, CD, DE, EF, FA

Aria = $\sum$ ariilor triunghiurilor ABC, ACD, ADE, AEF

Date de test:

a). `pentru poligonul A(1.00,1.00) B(-2.00,4.00) C(-4.00,-3.00)`

```
Aria = 13.500
Perimetrul = 17.926
```

b) `pentru poligonul A(2.98, 8.88) B(9.02, 5.98) C(13.16, 10.12) D(11.02, 14.00) E(6.04, 18.08)`

```
Aria = 68.732
Perimetrul = 33.119
```

Problema nr. 2.2

Se consideră polinomul cu coeficienți reali:

$$P(x) = a_0 \cdot x^n + a_1 \cdot x^{n-1} + \ldots + a_{n-1} \cdot x + a_n \qquad (2.1)$$

Să se calculeze, folosind schema lui Horner, valoarea polinomului în punctul **z** (de pe axa reală)

Relația de recurență pentru calculul valorii unui polinom într-un punct **z** este:

$$P_n(z) = P_{n-1}(z) \cdot z + a_n \quad \text{cu} \quad P_{-1}(z) = 0 \qquad (2.2)$$

(nu se vor folosi funcții recursive).

Se asociază unui polinom structura:

```
struct POLINOM {
    int n;                  // gradul polinomului
    double c[20];           // coeficienții polinomului
};
```

Se vor scrie și folosi funcții pentru:

- Citirea unui vector de numere reale;
- Afișarea polinomului luat în considerare
- Calculul valorii unui polinom de variabilă complexă cu coeficienți reali (varianta iterativă și nu recursivă) într-un punct.