**10.10.2024 Gimnaziu avansați**

**Tipul Inregistrare**

*Tip de date structurat, neomogen și anonim.*

## **Declararea unei structuri**

Sintaxa generală a declarării unui tip de date struct este următoarea:

struct [NUME\_STRUCTURĂ]

{

[TIP1 NUME\_CÂMP[, NUME\_CÂMP[, ...]];]

[TIP2 NUME\_CÂMP[, NUME\_CÂMP[, ...]];]

...

} [LISTA DE VARIABILE];

În acest mod s-a declarat un **tip de date** cu numele NUME\_STRUCTURA și eventual niște variabile de acest tip. În continuare NUME\_STRUCTURA poate fi folosit ca un tip de date oarecare, de exemplu pentru a declara variabile sau în lista parametrilor formali ai unei funcții.

Pentru exemplul dat mai sus, declararea va fi:

struct Persoana{

char nume[21], prenume[21], sex;

int varsta;

double salariu;

};

## **Variabile de tip structură**

În exemplul de mai sus s-a declarat o structură cu numele Persoana, dar nu s-a declarat nicio variabilă de acest tip. Pentru a declara variabile de tip Persoana putem să le adăugăm la declararea structurii:

struct Persoana{

char nume[21], prenume[21], sex;

int varsta;

double salariu;

} A, B;

(am declarat odată cu tipul de date Persoana și două variable A și B de acest tip), sau putem să le declarăm ulterior declarării tipului de date, astfel:

Această declarare este corectă în C++, dar nu este corectă în C:

Persoana A, B;

sau

Această declarare este corectă atât în C++ cât și în C:

struct Persoana A, B;

În declarările de mai sus avem următoarele:

* Persoana – este un tip de date, similar cu int sau char;
* A și B – sunt nume de variabile. Tipul acestor variabile este Persoana sau, mai explicit, struct Persoana;
* nume, prenume, sex, varsta, salariu – sunt numele unor câmpuri ale structurii Persoana. Ele pot fi utilizate într-un program doar dacă există variabile de tip Persoana!

Cea mai importantă operație cu structuri este **referirea unui câmp**.

## **Referirea unui câmp**

Se face cu ajutorul operatorului . (punct), numit și operator de acces direct la membrii unei structuri. Forma generală a acestei operații este:

variabila\_de\_tip\_structura.câmp

### Observații

* Rezultatul acestei operații este câmpul referit și este lvalue (poate fi supus unei atribuiri) ;
* Operația nu este comutativă;
* Despre accesul la membrii unei structuri referite de un pointer puteți citi [aici](https://www.pbinfo.ro/?pagina=articole&subpagina=afisare&id=7659" \l "pointeri-la-structuri);

### Exemplu

Considerăm următoarea secvență:

struct Persoana{

char nume[21], prenume[21], sex;

int varsta;

double salariu;

};

Persoana A, B;

Următoarele operații sunt corecte:

cin >> A.varsta;

strcpy(A.nume, "Popescu");

strcpy(B.nume, A.nume);

A.sex = 'M';

A.salariu = B.salariu + 125.80;

B.varsta ++;

Următoarele operații sunt greșite:

Persoana.nume

Persoana.A

varsta.persoana

## **Citirea și afișarea**

O variabilă de tip structură nu poate fi nici citită, nici afișată. Putem în schimb să realizăm aceste operații cu câmpurile structurii, folosind operatorul de access ..

CORECT!

cin >> A.nume >> A.prenume >> A.sex >> A.varsta >> A.salariu;

cout << A.nume << " " << A.prenume << " " << A.sex << " " << A.varsta << " " << A.salariu;

GREȘIT!

cin >> A;

cout << B;

## **Atribuirea**

Singura operație uzuală care se poate efectua cu o variabilă de tip structură este **atribuirea**:

A = B

unde A și B sunt două variabile de același tip structură. Atribuirea se face **bit cu bit** – fiecare bit din reprezentarea în memorie a variabilei B se copiază corespunzător în reprezentarea în memorie a variabilei A.

Uneori într-o structură putem avea câmpuri de alt tip structură. Exemplu:

Secvență C/C++

struct Data{

int zi, luna, an;

};

struct Persoana{

char nume[21], prenume[21], sex;

int varsta;

struct Data data\_angajarii;

double salariu;

};

struct Persoana A;

Secvență C++:

struct Data{

int zi, luna, an;

};

struct Persoana{

char nume[21], prenume[21], sex;

int varsta;

Data data\_angajarii;

double salariu;

};

Persoana A;

Am declarat doua tipuri de date: Data – pentru a memora o dată calendaristică, Persoana – pentru a memora informații despre o persoană și o variabilă A de tip Persoana. Variabila A are un câmp de tip Data, numit data\_angajarii. Prin intermediul acestuia avem acces la anul, luna și ziua angajării persoanei respective.

Următoarele operații sunt corecte.

A // de tip Persoana

A.data\_angajarii // de tip Data

A.data\_angajarii.an // de tip int

Următoarele operații sunt greșite. **DE CE?**

A.Data

A.Data.an

A.an.data\_angajarii

A.an.Data

A.salariu.an

## **Tablouri de structuri. Structuri cu câmpuri de tip tablou**

În C/C++ se pot declara tablouri cu elemente de tip structură și, de asemenea, o structură poate avea elemente de tip tablou (am văzut deja acest lucru în cazul tipului Persoana, cu câmpurile nume și prenume de tip șir de caractere).

Următoarea secvență C++ conține o asemenea situație:

struct Elev{

char nume[21], prenume[21], sex;

int numar\_note;

int note[10];

};

int n;

Elev V[28];

int main()

{

//....

for(int i = 0 ; i < n ; i ++)

{

cin >> V[i].nume >> V[i].prenume >> V[i].sex >> V[i].numar\_note;

for(int j = 0 ; j < V[i].numar\_note; j ++)

cin >> V[i].note[j];

}

//...

}

## **Funcții care lucrează cu structuri**

**De reținut:**

* o funcție C++ poate avea parametri de tip structură
* dacă dorim ca funcția să întoarcă un parametru modificat (cu câmpurile modificate), acesta trebuie transmis prin referință
* o funcție poate returna o valoare de tip structură

**Exemplu:**

Structura Punct memorează coordonatele întregi ale unui punct din plan:

struct Punct{

int x,y;

};

Funcția double Dist(Punct P) determină distanța de la origine la punctul P. Parametru este transmis prin valoare:

double Dist(Punct P){

return sqrt(P.x \* P.x + P.y \* P.y);

}

Funcția void Citire(Punct & P) citește coordonatele punctului P și întoarce coordonatele:

void Citire(Punct & P){

cin >> P.x >> P.y;

}

Funcția Punct Simetric(Punct P) returnează o structură cu coordonatele simetricului lui P față de origine:

Punct Simetric(Punct P){

Punct tmp;

tmp.x = - P.x, tmp.y = - P.y;

return tmp;

}

OLI2019-cls 8

OLI2020-cls 8

**Pbinfo.ro**

### #923 [FractiiMax](https://www.pbinfo.ro/probleme/923/fractiimax)

### [#922 Puncte](https://www.pbinfo.ro/probleme/922/puncte)

### #1013 [Aniversari](https://www.pbinfo.ro/probleme/1013/aniversari)

### #3481 [sort\_div](https://www.pbinfo.ro/probleme/3481/sort-div)

### #3164 [qclasa](https://www.pbinfo.ro/probleme/3164/qclasa)