**Variabile de tip înregistrare (record)**

O astfel de variabilă permite memorarea mai multor valori de tipuri DIFERITE.

Componentele (elementele, valorile) dintr-o astfel de variabilă NU se mai memorează pe indici, ci pe câmpuri.

Un câmp este un identificator format EXACT după aceleaşi reguli ca şi numele de variabile (adică trebuie să fie format DOAR din litere, cifre şi caracterul underscore, şi NU are voie să înceapă cu o cifră).

În definirea acestor tipuri de variabile avem posibilitatea să le definim şi declara pe loc, sau să definim doar o etichetă pentru tipul respectiv, prin care ulterior să declarăm de câte ori avem kef şi în ce loc avem kef, oricât de multe astfel de variabile.

Iată un exemplu prin care definim un tip de date "car" (de fapt "**car**" este o etichetă - aproape sinonim cu un tip de date) prin care reţinem câteva date despre o maşină, şi anume: model, an\_fabricatie, km\_parcursi, tip\_combustibil, cap\_cil :

struct **car**

{

char model[50];

int an\_fabricatie,km\_parcursi;

char tip\_combustibil;

double cap\_cil;

};

Pentru a declara ulterior variabilele a şi b de tipul **car** definit mai sus:

car a,b;

Obs: Putem combina definirea tipului de date cu declararea variabilelor, scriindu-le între

acolada închisă şi caracterul ; de după ea:

Obs: Putem chiar omite cuvântul car de după struct.

Pentru a accesa câmpurile unei astfel de variabile utilizăm expresia:

nume\_variabilă.nume\_câmp;

Fiecare câmp se comportă EXACT ca o variabilă de tipul respectiv.

Pe exemplul de mai sus, iată cum se iniţializează variabila **a** cu o maşină

Citroen C5, din 2008, capacitate cilindrică 2.5, pe motorină, km\_parcurşi 230000:

strcpy(a.model,"Citroen C5");

a.an\_fabricatie=2008;

a.km\_parcursi=230000;

a.tip\_combustibil='m';

a.cap\_cil=2.5;

Se pot defini evident şi vectori cu elemente de acest tip.

Ex:

car a[10];

fiecare element dintre cele 10, adică a[0], a[1], .. , a[9] este la rândul său de tip car.

Adică, după ce scriem a[indice] putem pune '.' şi continua cu câmpurile.

EX:

a[i].an\_fabricatie

Unul dintre marile avantaje ale acestui tip de date (adică ale struct-ului) este că putem face atribuiri între două variabile **de acelaşi tip**.

Ex:

b=a; (unde a şi b sunt cele de tip car declarate mai sus)

Caz în care se atribuie TOATE câmpurile lui a către variabila b.

Astfel, dacă declarăm o variabilă aux tot de tip car, putem interschimba între ele două variabile a şi b prin:

aux=a;a=b;b=aux;

**Struct-uri imbricate (nested)**

Putem defini un câmp al unui struct ca fiind de un alt tip, care şi el este tot un struct.

Ex:

Dacă avem o persoană pt. care dorim să reţinem o anumită dată calendaristică (de ex. data naşterii:

struct **data**

{ int zi,luna,an;};

struct pers {

char nume[30];

**data** data\_nasterii;

};

pers x;

strcpy(x.nume,"Matei");

x.data\_nasterii.zi=13;

x.data\_nasterii.luna=3;

x.data\_nasterii.an=2005;

Obs: Un stil de programare ceva mai vechi, derivat din C standard (ANSI) NU permite definirea tipului struct prin etichete, cum am făcut mai sus:

**struct eticheta { tip cimpuri; ...};**

ci se face astfel:

**typedef struct { tip cimpuri; ...} nume\_tipului\_de\_date; (eticheta)**

Ex:

**struct dreptunghi { int lung,lat;}; -** pe exemplul acesta "dreptunghi" este o

etichetă. Permite declararea ulterioară a unor variabile de acest tip. Cu toate

acestea, d.p.d.v. al limbajului, eticheta nu este chiar un tip de date.

**typedef struct { int lung,lat;} dreptunghi; -** pe exemplul acesta

"dreptunghi" este considerat ca un tip de date. Permite de asemenea

declararea ulterioară a unor variabile de acest tip.

Obs:Să reluăm struct-ul de mai sus:

struct **data**

{ int zi,luna,an;};

struct pers {

char nume[30];

**data** data\_nasterii;

} x;

El se poate declara şi direct imbricat, ca mai jos:

struct pers {

char nume[30];

struct **data**

{ int zi,luna,an;}

data\_nasterii;

} x;

Funcţionează fix la fel x.nume, x.data\_nasterii.zi,...

Însă, dacă declar aşa, tipul "data" nu va mai fi global, deci nu mai putem declara variabile de tip "**data**" în mod direct (dar totuşi chiar şi acestea pot fi declarate prin

pers::data variabilă )