Propunător: Ildegez Mihaela-Corina
Colegiul Național de Informatică *Spiru Haret* Suceava

Tipul înregistrare- struct

Cuprins:

Noțiuni introductive.

Să presupunem că ni se cere să întocmim o situație pentru școală, care să conțină următoarele date primare:

1. numele elevului
2. prenumele elevului
3. media generala
4. clasa

Observăm că pentru a memora aceste date cu cunoștințele acumulate până în prezent, ar trebui să utilizăm **patru tablouri unidimensionale**, două de tip șir de caractere, unul de tip real(float) și unul de tip întreg(int). Cu siguranță scrierea programului ar fi neplăcută. Apare astfel evidentă necesitatea existenței unui tip de dată cu ajutorul căruia să putem prelucra **date neomogene** ca cele din exemplul de mai sus. C++ ne pune la dispoziție un astfel de tip, numit **struct**, cu ajutorul căruia utilizatorul poate să își definească structura de date de care are nevoie.

Definirea unei structuri

Forma generală de definire a tipului struct:

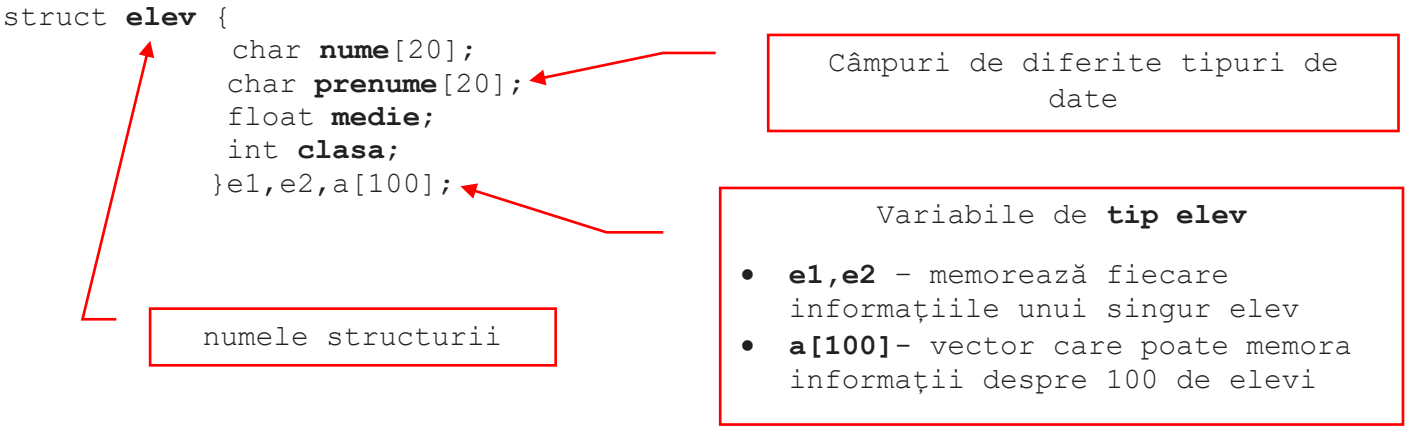
```
struct [<nume structură>] {
                           tip1 <nume câmp> [,<nume câmp>, …];
                           tip2 <nume câmp > [,<nume câmp >, …];
                           ....
                           tipn <nume câmp > [,<nume câmp >, …];
                           } [listă variabile];
```

Unde:

- **nume structură** - desemnează numele sub care va fi utilizat noul tip de date în program, care va fi unul structurat. Numele structurii poate lipsi din definirea ei, iar în acest caz spunem că avem o structură anonimă(acest lucru este posibil dar nerecomandat)
- **<tip1>…<tipn>** - reprezintă tipurile de date stabilite de utilizatori pentru câmpurile(articolele) din care se compune structura
- **<nume câmp>** - desemnează articolelor care compun structura
- `[listă variabile]` - reprezintă variabilele declarate care vor fi de ti struct

Tipul struct descrie doar modul cum sunt organizate datele, asemenea tipurilor predefinite, nu rezervă o zonă de memorie în care datele se păstrează. Pentru a memora datele **este obligatorie declararea unei variabile** de tip <**nume structură**>.

Exemplul : definim următoarea structură:



Declararea unei variabile de tip struct

Declararea variabilelor de tipul structurii definite se poate realiza în două moduri:

1. După cum putem observa din forma generală, variabilele se pot declara după definirea structurii. În acest caz structura poate fi anonimă, adică <nume structură> poate să lipsească.

2. Oriunde în program, în același mod cu declararea variabilelor pentru tipurile predefinite:

<**nume structură**> <nume var1>[, <nume var2>, …,<nume varn>];

În acest caz **<nume structură>** este obligatoriu, la definirea structurii. Definirea unui tip structură poate fi plasată oriunde în program, acest tip va fi cunoscut din momentul definirii lui până în momentul încheierii structurii în interiorul căreia a fost definit.

Modul de adresare(accesul la câmpurile structurii)

Cum ne adresăm unui anumit membru al unei structuri, știind că avem o singură variabilă care cuprinde toate câmpurile structurii? Pentru a putea accesa un anumit câmp din interiorul unei structuri vom folosi operatorul de selecție directă **“.”** , acest **operator are prioritate maximă**. Pentru exemplul dat:

```
struct elev {
              char nume[20];
              char prenume[20];
              float medie;
              int clasa;
             }e, a[100];
```

Ne vom adresa unui câmp din variabila:

e	variabila a – elementul din poziția i
e.nume	a[i].nume
e.prenume	a[i].prenume
e.medie	a[i].medie
e.clasa	a[i].clasa

Operații cu variabile declarate de tip structura

1. Operația de atribuire: atribuirea este permisă între două variabile declarate de tip struct, doar dacă ambele variabile sunt definite de același tip

2. Operația de citire/scriere: citirea, respectiv afișarea unei variabile definite de tip struct este permisă doar câmp cu câmp.

3. **Operații specifice** pentru tipul fiecărui câmp din structură.

Exemplu: fie următoarea declarare de tipuri și variabile:

```
struct elev {
             char nume[20];
             char prenume[20];
             float medie;
             int clasa;

            }e, a[100];
```

Operații specifice:

e	variabila a – elementul din poziția i
Citirea variabilei de tip **elev**	

`cin >> e.nume;`
`cin >> e.prenume;`
`cin >> e.medie;`
`cin >> e.clasa;`
	`cin >> a[i].nume;`
	`cin >> a[i].prenume;`
	`cin >> a[i].medie;`
	`cin >> a[i].clasa;`

Afișarea variabilei de tip **elev**	

`cout >> e.nume;`
`cout >> e.prenume;`
`cout >> e.medie;`
`cout >> e.clasa;`
	`cout >> a[i].nume;`
	`cout >> a[i].prenume;`
	`cout >> a[i].medie;`
	`cout >> a[i].clasa;`

Atribuirea între două variabile de tip elev	
`strcpy(e.nume,"Ionescu");` `e=a[1];`	`a[1]=a[3];` `a[i]=e;`

Înregistrări imbricate

Să presupunem că pe lângă datele **elevului** din exemplul precedent trebuie să mai adăugăm situației noastre și **data nașterii** elevului. O dată calendaristică se compune din **3** câmpuri distincte, notate: **zi, lună, an**. Vom putea defini datele unui elev folosind **două structuri**, astfel:

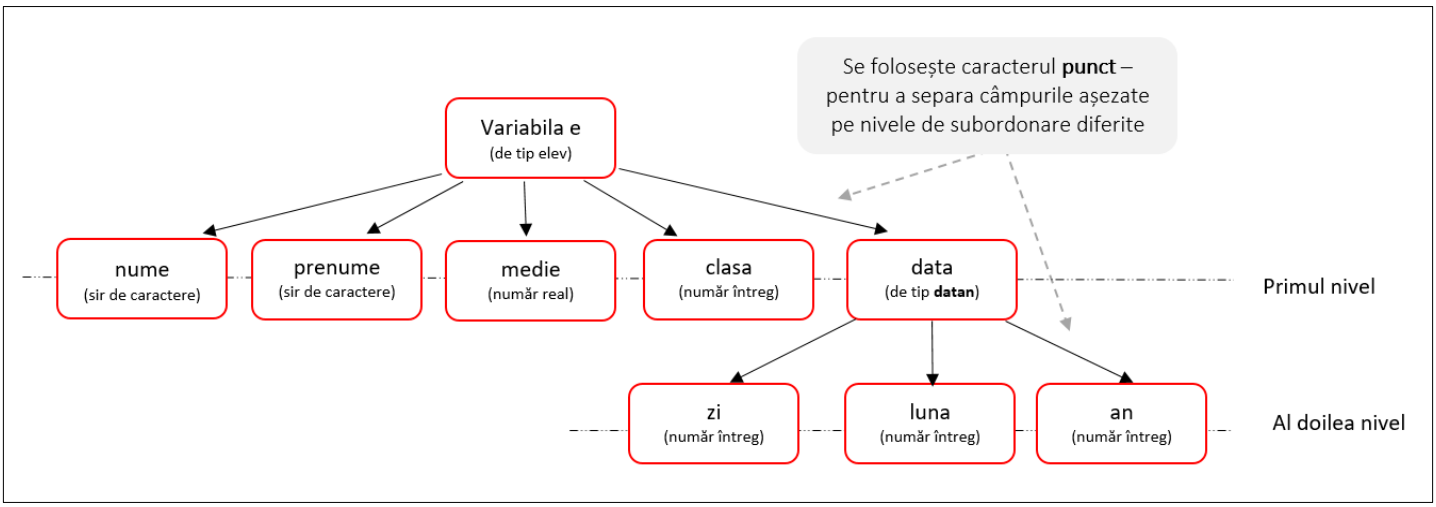
```
struct datan{
             int zi;
             int luna;
             int an;
             };
struct elev{
             char nume[20];
             char prenume[20];
             float medie;
             int clasa;
             datan data;
       }e;
```

Tipul **datan** se utilizează pentru memorarea Unei date calendaristice in forma **zi, luna, an.**

Tipul **elev** se utilizează pentru memorarea **numelui, prenumelui, media, clasa** și **data nașterii** unui elev

Variabilă de tip elev

În exemplul de mai sus avem două structuri imbricate – câmpul data din structura elev este de tip datan, care la rândul lui este de tip struct. Modul de acces la fiecare câmp din exemplul dat, poate fi reprezentat grafic în următoarea diagramă:



Variabila	Câmp	Modalitate de acces
e	nume	e.nume
	prenume	e.prenume
	medie	e.medie
	clasa	e.clasa
	zi	**e.data.zi**
	luna	**e.data.luna**
	an	**e.data.an**

Observații:

- Pot exista mai multe nivele de imbricări
- Operațiile permise sunt în funcție de tipul fiecărui câmp

Structuri care conțin câmpuri de tip tablouri unidimensionale

Câmpurile unei structuri pot fi declarate de orice tip de date, fie prestabilit, fie declarat anterior de utilizator. Deci într-o structură putem avea si câmpuri care sunt tablouri unidimensionale(vectori). Un exemplu poate fi:

```
struct Lista{
            int n;
            int a[100];
            }L;
```

Variabila	Câmp	Modalitate de acces
L	n	**L.n**
	a[1]	**L.a[1]** -elementul de pe poziția 1 din vectorul a
	a[2]	**L.a[2]** -elementul de pe poziția 2 din vectorul a
	...	...
	a[i]	**L.a[i]** -elementul de pe poziția i din vectorul a
	...	...
	a[n]	**L.a[n]** -elementul de pe poziția n din vectorul a

Spre exemplu operația de citire/afișare a vectorului poate fi scrisă astfel:

Citire	Afișare pe ecran
... cin>> **L.n**; for(int i=1;i<=n;i++) cin>> **L.a[i]**;	for(int i=1;i<=**L.n**;i++) cout << **L.a[i]** << " "; cout<<endl;

...	

Vectori de structuri

Așa cum știm din definiția tablourilor unidimensionale toate elementele sale au același tip. In acest caz tipul elementelor unui vector poate fi un tip structurat. Dacă avem următoarea declarare de tip:

```
struct datan{
             int zi;
             int luna;
             int an;
             };

struct elev{
            char nume[20];
            char prenume[20];
            float medie;
            int clasa;
            datan data;
       }a[101];          ///vector care poate memora 100 de elevi, indexat de la 1
int n;
```

Variabila	Poziția	Câmp nivel 1	Câmp nivel 2	Modalitate de acces
a	i $i \in [1,n]$	**nume**		**a[i].nume** – numele elevului i
		prenume		**a[i].prenume** – prenumele elevului i
		medie		**a[i].medie** – media elevului i
		clasa		**a[i].clasa** – clasa elevului i
		data		
			zi	**a[i].data.zi** – ziua nasterii elevului i
			luna	**a[i].data.luna** – luna nasterii elevului i
			an	**a[i].data.an** – anul nasterii elevului i

Spre exemplu operația de citire/afișare a vectorului poate fi scrisă astfel:

Citire	Afișare pe ecran

...
cin>> **n**;
for(int i=1;i<=n;i++)
  cin>> **a[i].nume** >>
  a[i].prenume >>
  a[i].medie >>
  a[i].clasa >>
  a[i].data.zi >>
  a[i].data.luna >>
  a[i].data.an;
...
	...
	for(int i=1;i<=n;i++)
	  cout << **a[i].nume**<< " " <<
	  a[i].prenume<< " " <<
	  a[i].medie << " " <<
	  a[i].clasa << " " <<
	  a[i].data.zi << " " <<
	  a[i].data.luna << " " <<
	  a[i].data.an<< " "<<endl;
	...

Dacă spre exemplu dorim să ordonăm descrescător elevii după medie, folosind o metodă de sortare putem scrie:

```
for(int i=1; i < n; i++)
   for(int j=i+1; j<=n; j++)
     if(a[i].medie < a[j].medie)
       { elev aux;
         aux=a[i];
         a[i]=a[j];
         a[j]=aux
```

```
        }
```