Fundamentele programării

Curs 7

Structuri

Elena BAUTU & Dorin-Mircea POPOVICI
{ebautu,dmpopovici}@univ-ovidius.ro
Web: http://disciplinele.ml

Cuprins

- Structuri
  - Definiție, declarare, inițializare, asignare, exemple
  - Masive de structuri
  - Pointeri la structuri
- Funcții ce manevreaza structuri
- Structuri cu autoreferire – liste
- Dincolo de structuri ...

Structuri

Structuri – De ce?

- De ce as avea nevoie de o “structura”?
- Masivele nu sunt suficiente?

- Manevrati o colectie de puncte/primitive 3D (ca sa va creati propriul motor grafic).
- Gestionati stocul de produse al firmei voastre.
- Implementati o forma electronica a unui catalog asociat unui examen de semestru.

Structuri – Cum ajungem la ele?

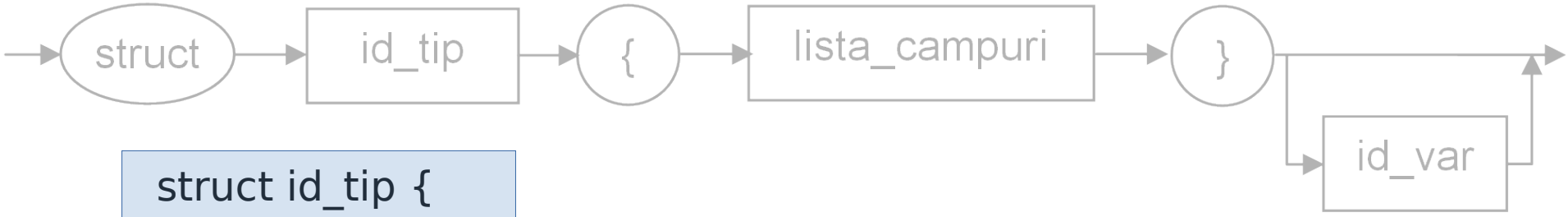
- Concept: Punct3D
- Date: coordonatele x, y, z
- Operatii: creare, modificare, copiere, stergere

- Concept: Stoc
- Date: produs, cantitate, pretU, conditii de depozitare, timp de asteptare
- Operatii: receptie, livrare, incasare, gestionare, calcul sold

- Concept: Catalog
- Date: disciplina, titular, credite, studenti, note
- Operatii: gestionare note, afisare rezultate

Structuri

- O structură este un tip de date care grupează date de tipuri diferite sub un singur nume.
  - Structura este ***eterogenă***
  - un element al unei structuri se numeşte ***membru*** sau ***câmp***
  - fiecare câmp are un tip de date propriu
    - informaţiile au tipuri şi locaţii diferite
- O structură se declară cu cuvântul cheie **struct**, astfel:




```
struct id_tip {
   tip1 membru1;
   tip2 membru2;
   ...
   tipn membrun;
};
```

Structuri – exemple

```
struct Punct3D {
        float x,y,z;
};
```

```
struct Grupa{
        char nume[3];
        int an;
};
```

```
struct Timp {
        int ora;
        int minut;
        int secunda;
};
```

```
struct Student  {
        char nume[64];
        struct Grupa g;
        int anNastere;
};
```

Variabile tip structură - declarare

- Alocarea de memorie se face în momentul *definirii* unor variabile de tipul structurii
- *Declararea* unei variabile de tipul structură se poate face:
  1. la momentul declarării structurii

```
struct id_tip {
        tip_1 membru_1;
        tip_2 membru_2;
        ...
        tip_n membru_n;
} id_var;
```

  2. după declararea structurii

```
id_tip id_var;
```

```
struct complex { //cazul 1
    double re, im;
} a, b;

struct timp { //cazul 1
     int ora, minut, secunda;
} moment;
```

```
struct complex { //cazul 2
     double re, im
};
struct complex a, b;
```

Variabile tip structură - declarare

- Cuvântul cheie *typedef*

  typedef nume_vechi nume_nou;

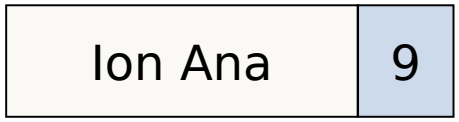
- Exemplu

  typedef struct complex Complex;

  //apoi putem defini variabile
  Complex a, b;

Accesarea campurilor

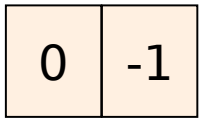
- Accesarea unui camp: id_var.numecâmp




```
struct student {
  char nume[20];
  int nota;
} s1;
```

- nume structura: student
- nume variabila: s1
- număr câmpuri: 2
- accesare câmp:

```
s1.nota = 10;
strcpy(s1.nume, “Popa Catalin”);
```




```
struct complex {
  double re;
  double im;
} c1;
```

- nume structura: complex
- nume variabila: c1
- număr câmpuri: 2
- accesare câmp:

```
c1.im = -1;
```

Iniţializarea structurilor

- Structurile pot fi iniţializate similar cu masivele

```
struct complex {
        double re, im
  } ;

  struct complex i = {0,1};
```

```
struct grupa{
      char nume[3];
      int an;
}
struct grupa g1 = {“b1”, 1};
```

- Expresiile cu care se inițializează sunt *constante*
- Dacă sunt date insuficiente la inițializare, restul variabilelor membru se inițializeaza cu 0.
- Este *permisă* inițializarea prin asignare ori prin apelul unei funcții ce returneaza tipul potrivit de structură.

Masive de structuri

- Instrucțiunea

```
Punct3D v[N];
```

declară un vector de structuri Punct3D, deci alocă memorie pentru N structuri Punct3D

- Fiecare element al vectorului este o structură;
- Puntem accesa membrii primului element:

```
v[0].x=-1, v[0].y=-1; v[0].z=0;
```

Masive de structuri

```
struct Punct3D {
      float x,y,z;
} v[3];
```

```
v[0].x=v[0].y=v[0].z=0;
v[1]=v[0]; //copierea se realizeaza bit by bit
v[2].x=v[1].x+15;
v[2].z=v[0].z-v[2].x+30;
```

Masive de structuri

```
struct Punct3D {
        float x,y,z;
} v[3]={{1,0,0},{0,1,0}};
```

Exercitiul 6.1

- Scrieti un program care citeşte informaţiile unei grupe de studenţi, le sortează descrescător după notă şi afişează rezultatele.

- Scrieti un program care determina dreptunghiul de arie minima, aliniat la sistemul de axe de coordonate si care cuprinde un set de puncte

```
#include <stdio.h>
typedef struct student
{
char nume[20];
char prenume[20];
int nota;
} Student;

int main()
{
Student student[25], aux;
int i,j,n;

printf("Numar studenti:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Nume si prenume student
   %d:\n", i+1);
scanf("%s%s",student[i].nume,
   student[i].prenume);
printf("Nota studentului %s %s:\
   n", student[i].nume,
   student[i].prenume);
scanf("%d",&student[i].nota);
}
```

```
for(i=0;i<n;i++)
   for(j=i+1;j<n;j++)
         if(student[i].nota <
               student[j].nota)
         {
            aux=student[i];

   student[i]=student[j];
               student[j]=aux;
        }
printf("Rezultatele grupei:\n");
for(i=0;i<n;i++)
printf("%s %s\t%d\n",
   student[i].nume,
   student[i].prenume,
   student[i].nota);
}
```

Structuri imbricate (nested)

- typedef struct student {
  char nume[64];
  struct grupa g;
  int anNastere;
  } Student;

Student s = {“Vasilache Anton”, {“b1”, 1}, 1998};

s.g.an = 2;

Pointeri la structuri

- Declararea unui pointer la o structura

```
struct complex{ float re, float im} *pc;
```

Sau

```
Complex * pc;
```

- Accesul la variabilele membru:

```
(*pc).re, (*pc).im // parantezele () sunt obligatorii
//alternativ, se folosește operatorul ->
pc->re , pc->im
```

Utilitate: când o structură (“mare”) este pasată ca parametru unei funcții, este mai eficient sa se trimită un pointer (în loc să se copieze întreaga structură)

Pointeri la structuri – Exemple

```
Complex z = {0, 1}, *pz;
pz = &z;
printf("%d+%d*i", z.re, z.im);
printf("%d+%d*i", pz->re, pz->im);

pz->re = 1;
printf("%d+%d*i", z.re, z.im);
printf("%d+%d*i", pz->re, pz->im);
printf("%d+%d*i", (*pz).re, (*pz).im);
```

Atribuire de structuri

```
Complex z1 = {1,1}, z2, z3, *pz;

// copierea se face membru cu membru
z2 = z1;
printf("%d+%d*i", z2.re, z2.im);

pz = &z2;
printf("%d+%d*i", (*pz).re, (*pz).im);
pz -> re=2;
z3 = *pz;
printf("%d+%d*i", z3.re, z3.im);
```

Atribuire de structuri (cont.)

- **Dacă structura conține vectori, aceștia se copiază (implicit) element cu element**

```
struct student{
   char nume[20]; int nota;
   } s1={“Ana”, 10}, s2;
   s2=s1;
   printf(“%s:%d”,s1.nume, s1.nota);
   printf(“%s:%d”,s2.nume, s2.nota);
   strcpy(s2.nume,  “Arabela”);
   printf(“%s:%d”,s1.nume, s1.nota);
   printf(“%s:%d”,s2.nume, s2.nota);
```

Functii ce manevreaza structuri

Funcție cu parametru structura

- O funcție poate primi ca *parametru* o *structură:*

```
typedef struct student{
        char nume[20]; int nota;
} Student;
```

- Transferul unei structuri la functii se realizează prin **valoare**

Sa scriem doua functii, una care afiseaza continutul unui student si o alta care incarca numele si nota unui student

```
void afiseaza(Student s){
    printf("%s:%d\n", s.nume, s.nota);
    }
```

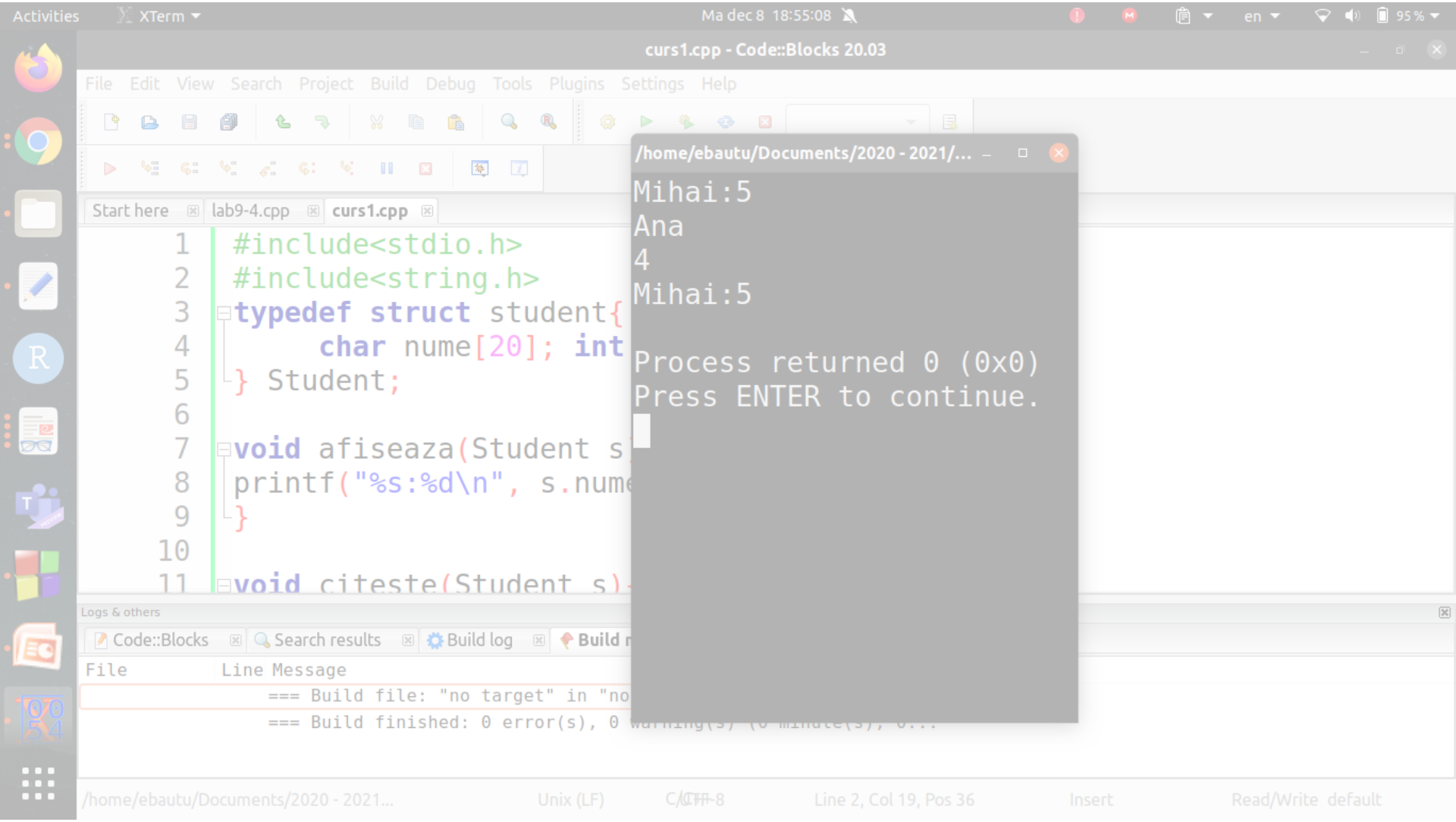
```
void citeste(Student s){
    scanf("%s", s.nume);
    scanf("%d", &s.nota);
    }
```

Funcție cu parametru structura - cont

- Sa punem la treaba functiile scrise si urmarim efectele aplicarii lor:

```
int main(){
    Student st;
    strcpy(st.nume, “Mihai”);
    st.nota = 5;

    afiseaza(st);
    citeste(st); //introducem numele Ana si nota 4
    afiseaza(st);
}
```


Activities
XTerm
Ma dec 8 18:55:08
en
95 %
curs1.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Tools Plugins Settings Help
/home/ebautu/Documents/2020 - 2021/...
Mihai:5
Ana
4
Mihai:5
Process returned 0 (0x0)
Press ENTER to continue.
Start here
lab9-4.cpp
curs1.cpp
#include<stdio.h>
#include<string.h>
typedef struct student{
char nume[20]; int
} Student;
void afiseaza(Student s
printf("%s:%d\n", s.nume
}
void citeste(Student s)
Logs & others
Code::Blocks
Search results
Build log
File
Line Message
=== Build file: "no target" in "no
=== Build finished: 0 error(s), 0
/home/ebautu/Documents/2020 - 2021...
Unix (LF)
Line 2, Col 19, Pos 36
Insert
Read/Write default

Funcție cu parametru pointer la structura

- O funcție poate primi ca *parametru* un *pointer* la o structură
- Sa scriem variantele celor doua functii, avand de data aceasta ca parametru un pointer catre un student

```
void afiseaza(Student *ps){
  printf("%s:%d\n",ps->nume, ps->nota);
}
```

```
void citeste(Student *ps){
   printf("Dati un nume si o nota");
   gets(ps->nume);
   scanf("%d", &ps->nota);
}
```

Funcție cu parametru pointer la structura - cont

- Sa punem la treaba functiile scrise si urmarim efectele aplicarii lor:

```
int main(){
    Student st;
    strcpy(st.nume, “Mihai”);
    st.nota = 5;

    afiseaza(&st);
    citeste(&st); // numele George si nota 10
    afiseaza(&st);
}
```

Funcție ce returneaza o structura

- O funcție poate *întoarce* ca rezultat o structură

```
Student citeste1(){
        Student s;
        puts(“Introduceti numele studentului”);
        gets(s.nume);
        puts(“Introduceti nota studentului”);
        scanf(“%d”, &s.nota);
        return s;
}
int main(){
        Student s1=citeste1();
        …
}
```

Exercitiul 6.2

/* Scrieţi o structură care să definească un punct de coordonate (x,y), cu x,y numere intregi. Cititi apoi informatii despre doua puncte de la tastatura si afisati distanta dintre ele. */

```
typedef struct punct{
	int x, y;
} Punct ;
int main{
	// definiti doua puncte:
	Punct A, B;
	// Citiţi informaţiile celor doua puncte de la tastatura:
	printf(“Introduceti coordonata x a lui A:”); scanf(“%d”, &A.x);
	printf(“Introduceti coordonata y a lui A:”); scanf(“%d”, &A.y);
```

Exercitiul 6.2 - cont

```
printf(“Introduceti coordonata x a lui B:”); scanf(“%d”, &B.x);
printf(“Introduceti coordonata y a lui B:”); scanf(“%d”, &B.y);

//calc. distanta dintre cele puncte
 double distanta = sqrt(((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y)));
    printf("Distanta intre cele doua puncte este %lf\n", distanta);

    return 0;
}
```

Exercitiul 6.3

- Modificați programul așa încât să includeți funcții pentru citirea informațiilor despre o structura de la tastatura, respectiv pentru afișarea informațiilor despre o stuctura Punct pe ecran:
  - Implementați o varianta de citire în care funcția de citire modifică o structură adresată printr-un pointer primit ca parametru
    - void citestePunct1(Punct* pp);
  - Implementați o variantă de citire în care funcția returnează o structură
    - Punct citestePunct2();
  - void afiseazaPunct(Punct p);
- Implementati o funcție pentru calculul distanței între doua puncte
  - double distanta(Punct p1, Punct p2);
- Implementati functii corespunzatoare transformarilor geometrice translatie, si scalare a unui punct, precum si de rotatie a unui punct in jurul altui punct cu un unghi α.

Exercitiul 6.3 - sugestie

```
typedef struct punct{
            int x; int y;
} Punct ;
void citestePunct1(Punct* pp);
Punct citestePunct2();
void afiseazaPunct(Punct p);
double distanta(Punct p1, Punct p2);
int main(){
   Punct p1, p2;
   citestePunct1(&p1);
   p2=citestePunct2();
   afiseazaPunct(p1);
   afiseazaPunct(p2);
   printf("Distanta intre cele doua
puncte este %f", distanta(p1, p2));
}
```

```
void citestePunct1(Punct* pp){
   puts(“Coordonatele punctului:”);
   scanf(“%d%d”, &pp->x, &pp->y);
}
Punct citestePunct2(){
  Punct p;
  puts(“Coordonatele punctului:”);
  scanf(“%d%d”, &p.x, &p.y);
  return p;
}
void afiseazaPunct(Punct p){
   printf(“Punctul de coordonate (%d, %d)\
n”, p.x, p.y);
}
double distanta(Punct p1, Punct p2){
   return  sqrt((pow((p1.x-
p2.x),2)+pow((p1.y-p2.y),2)));
}
```