Structuri. Uniuni. Aplicaţie: Matrice rare

Resposabili:Responsabili:

- Dorinel Filip (CA 2016-2020) [mailto:ion_dorinel.filip@cti.pub.ro]
- Darius Neațu (CA 2019-2020) [mailto:neatudarius@gmail.com]
- Mihaela Vasile [mailto:mihaela.vasile@gmail.com]

Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- organizeze datele din rezolvarea unei probleme complexe în structuri şi uniuni;
- optimizeze scrierea funcţiilor prin minimizarea numărului de parametri şi prin utilizarea structurilor ca parametri returnaţi de funcţie;
- distingă diferenţa dintre o structură şi o uniune;
- evite utilizarea greşită a structurilor.

Noţiuni teoretice

Structuri

Structurile sunt tipuri de date în care putem grupa mai multe variabile eventual de tipuri diferite (spre deosebire de vectori, care conţin numai date de acelasi tip). O structură se poate defini astfel:

```
struct nume_structura {
  declaratii_de_variabile
};
```

Definiția unei structuri poate fi urmată imediat de declararea de variabile de acel tip, forma cea mai generală fiind:

```
struct [structure tag] {

   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

Unde între [ ] se află câmpurile opționale.

Exemple:

```
struct student {
  char nume[40];
  int an;
  float medie;
};
```

Variabilele declarate in interiorul structurii se numesc “campurile” structurii:

- nume;
- an;
- medie.

```
struct complex { /* pentru memorarea unui număr complex cu dublă precizie */
  double re;
  double im;
};
```

Declararea şi iniţializarea unor variabile de tip structură se poate face astfel:

```
struct student s1 = {"Popescu Ionel", 3, 9.25};
struct complex c1, c2;
struct complex v[10];
```

Pentru simplificarea declaraţiilor, putem asocia unei structuri un nume de tip de date:

```
typedef struct student Student;
...
Student s1, s2, s3;
```

Redenumirea de tip de mai sus poate fi inclusă și în definirea structurii, caz în care putem folosi - din prima - atât `struct Student` cât și tipul de date rezultat din typedef:

```
typedef struct student {
  char nume[40];
  int an;
  float medie;
} Student;

int main() {
   /* Ambele declaratii de mai jos sunt valide */
   struct student s1;
   Student s2;
}
```

Accesul la membrii unei structuri se face prin operatorul ”.”:

```
s1.an = 3;
```

În cazul pointerilor la structuri, accesul la membri se poate face astfel:

```
Student *stud = (Student *)malloc(sizeof(Student));
(*stud).medie = 9.31;
/* altă modalitate mai simplă și mai des folosită: */
stud->medie = 9.31;
```

Atribuirile de structuri se pot face astfel:

```
struct complex n1, n2;
...
n2 = n1;
```

Prin această atribuire se realizează o copiere bit cu bit a elementelor lui n1 în n2.

Alt exemplu de utilizare: După cum se vede mai jos trebuie facută diferenta când definim un tip şi cand declaram o variabila de tip struct sau typedef struct.

```
typedef struct {
  int data;
  int text;
} S1; // este un typedef pentru S1, functional in C si C++

struct S2 {
  int data;
  int text;
}; // este un typedef pentru struct S2
```

```
struct {
  int data;
  int text;
} S3;
// este o declaratie a lui S3, variabila de tip struct nu defineste un tip
// spune compilatorului sa aloce memorie pentru variablia S3

typedef struct S4 {
  int data;
  int text;
} S4; // este un typedef atât pentru struct S4, cât și pentru S4


int main() {
  // ce se intampla la declarare variabile de tip S1,S2,S3
  S1 mine1; // este un typedef si va merge
  struct S2 mine2; // este un typedef si va merge
  S2 mine22; // S2 NU este un typedef si NU va merge
  S3 mine3; // nu va merge pt ca S3 nu este un typedef.
  struct S4 mine4; // este un typedef și va merge
  S4 mine4; // este un typedef și va merge

  // ce se intampla la utilizarea ca variabile a S1,S2,S3
  S1.data = 5; // da eroare deoarece S1 este numai un typedef.
  struct S2.data = 5; // da eroare deoarece S2 este numai un typedef.
  S3.data = 5; // merge doarece S3 e o variabila
  return 0;
}
```

Dacă declaraţi pointeri la structuri, nu uitaţi să alocaţi memorie pentru aceştia înainte de a accesa câmpurile structurii. Nu uitaţi să alocaţi şi câmpurile structurii, care sunt pointeri, înainte de utilizare, dacă este cazul. De asemenea fiţi atenţi şi la modul de accesare al câmpurilor.

Diferenţa dintre copierea structurilor şi copierea pointerilor

Pentru exemplificarea diferenţei dintre copierea structurilor şi copierea pointerilor să considerăm urmatorul exemplu:

```
struct exemplu {
  int n;
  char *s;
}
struct exemplu s1, s2;
char *litere = "abcdef";
s1.n = 5;
s1.s = strdup(litere);
s2 = s1;
s2.s[1]='x';
```

După atribuirea s2 = s1;, s2.s va avea o valoare identică cu s1.s. Deoarece s este un pointer (o adresă de memorie), s2.s va indica aceeaşi adresa de memorie ca şi s1.s. Deci, după modificarea celui de-al doilea caracter din s2.s, atat s2.s cât si s1.s vor fi axcdef. De obicei acest efect nu este dorit şi nu se recomandă atribuirea de structuri atunci cand acestea contin pointeri.

Totuşi, putem atribui ulterior lui s2.s o altă valoare (o altă adresă), iar ca urmare a acestei operaţii, stringurile vor fi distincte din nou.

Un alt caz (diferit de cel expus anterior) este cel al atribuirii aceleiaşi structuri către două variabile pointer diferite:

```
struct exemplu {
  int n;
  char * s;
}
struct exemplu s1;
struct exemplu *p1;
struct exemplu *p2;
```

```
p1 = &s1;
p2 = &s2;
```

În acest caz observăm că din nou p1→s şi p2→s indică către acelaşi şir de caractere, dar aici adresa către şirul de caractere apare memorată o singura dată; spre deosebire de cazul anterior, dacă modificăm adresa din p2→s, ea se va modifica automat şi în p1→s.

Uniuni

Uniunile sunt asemănătoare structurilor, dar lor li se rezervă o zonă de memorie ce poate conţine, la momente de timp diferite, variabile de tipuri diferite. Sunt utilizate pentru a economisi memoria (se refoloseşte aceeaşi zonă de memorie pentru a stoca mai multe variabile).

Uniunile se pot declara astfel:

```
union numere {
  int i;
  float f;
  double v;
};              /* se poate utiliza si typedef... */

union numere u1, u2;
```

Când scriem ceva într-o uniune (de exemplu când facem o atribuire de genul u1.f = 7.4), ceea ce citim apoi trebuie să fie de acelaşi tip, altfel vom obţine rezultate eronate (adică trebuie să utilizam u1.f, nu u1.v sau u1.i). Programatorul trebuie să ţină evidenţa tipului variabilei care este memorată în uniune în momentul curent pentru a evita astfel de greşeli. Operaţiile care se pot face cu structuri se pot face şi cu uniuni; o structura poate conţine uniuni şi o uniune poate conţine structuri.

Exemplu:

```
#include <stdio.h>
#include <stdlib.h>

typedef union {
  int Wind_Chill;
  char Heat_Index;
} Condition;

typedef struct {
  float temp;
  Condition feels_like;
} Temperature;

int main() {

  Temperature *tmp;
  tmp = (Temperature *)malloc(sizeof(Temperature));

  printf("\nAddress of Temperature = %u", tmp);
  printf("\nAddress of temp = %u, feels_like = %u",
                &(*tmp).temp,&(*tmp).feels_like);
  printf("\nWind_Chill = %u, Heat_Index= %u\n",
                &((*tmp).feels_like).Wind_Chill,
                &((*tmp).feels_like).Heat_Index);

  return 0;
}
```

La rulare va afişa:

```
Address of Temperature = 165496
Address of temp = 165496, feels_like = 165500
Wind_Chill = 165500, Heat_Index= 16550
```

Exercitii laborator CB/CD

Codul sursa se gaseste aici [http://swarm.cs.pub.ro/~gmuraru/PC/lab10.c] Primul exercitiu presupune modificarea/adaugarea de instructiuni unui cod existent pentru a realiza anumite lucruri. In momentul actual programul numara cate persoane au acelasi prenume.

Cerinte:

- Sa se afiseze cate persoane s-au nascut in acelasi an
- Sa se sorteze persoanele dupa nume

Următoarele două probleme vă vor fi date de asistent în cadrul laboratorului.

Tutorial folosire checker laborator [https://ocw.cs.pub.ro/courses/programare/checker]

Exerciţii de laborator

1. [2p] O matrice rară (cu circa 90% din elemente 0) este păstrată economic sub forma unei structuri, care conţine următoarele câmpuri:
   - int L,C - numărul de linii/coloane al matricei rare
   - int N - numărul de elemente nenule
   - int LIN[] - vectorul ce păstrează liniile în care se află elemente nenule
   - int COL[] - vectorul ce păstrează coloanele în care se află elemente nenule
   - float X[] - vectorul ce păstrează elementele nenule
2. Să se definească funcţii pentru:
   - [1p] citirea unei matrice rare
   - [1p] afişarea unei matrice rare (ca o matrice, cu tot cu zerouri)
   - [2p] adunarea a două matrice rare.
   - **Observatii** :
   - Vectorii LIN, COL, X vor avea fiecare cate N elemente.
   - Elementul nenul cu valoare X[index] se afla pe linia LIN[index] si coloana COL[index].
   - Elementele vor fi citite in ordinea crescatoare a liniilor, iar elementele de pe aceeasi linie in ordinea crescatoare a coloanelor.
3. Un polinom este reprezentat printr-o structură care conţine gradul polinomului şi tabloul coeficientilor. Să se definească funcţii pentru:
   - [1p] citirea unui polinom
   - [1p] afişarea unui polinom
   - [2p] adunarea/scăderea a două polinoame

Bonus

1. [3p] În registrul `Uni-Struct`, fiecărui elev îi corespunde o înregistrare care reține:

   - numele acestuia (maxim 50 de caractere),
   - clasa în care acesta se află (o valoare de tipul `unsigned int`),
   - rezultatul obținut la materia cea mai apropaiata (ca semnificație) de programare.

Tipul rezultatului diferă în funcție de clasa în care elevul se află, astfel:

- elevii din clasele primare (1-4), primesc un calificativ (FB, B, S, I) reprezentat ca un șir maxim 9 caractere;

- elevii din toate celelalte clase primesc note, care vor fi reținute cu ajutorul unui câmp de tip `float`.

Scrieți un program care citește de la tastatură astfel de înregistrări, până la inserarea șirului de caractere "`stop`" în locul numelui unuia dintre elevi, apoi afișează toți studenții din registru, alături de toate informațiile despre ei, în ordinea lexicografică a numelor cu care sunt înregistrați.

Folosiți un vector de structuri pentru a stoca toate informațiile despre elevi. Faceți eficient reținerea unei singure valori pentru rezultatul parcurgerii materiei în cadrul structurii.

bonus.in

```
Ionut Popescu
10
7.4
Maria Almasan
3
FB
Alex Grigorescu
12
10
stop
```

bonus.out

```
Alex Grigorescu
12
10
Ionut Popescu
10
7.4
Maria Almasan
3
FB
```

Referinţe

- The GNU Programming Tools - Data structure [http://crasseux.com/books/ctutorial//Data-structures.html#Data%20structures]
- The C Book - Unions [http://publications.gbdirect.co.uk/c_book/chapter6/unions.html]