UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structuri de Date și Algoritmi

Struct in C

Profesor: Maria Guțu

Struct in C

Struct in C:

Struct in C – semnatică, sintaxă, exemple de implementare.

Struct in C

Structurile sunt un tip de date compus care permite gruparea diferitor tipuri de date sub o singură entitate logică într-un singur bloc de memorie. Ele sunt folosite pentru a reprezenta obiecte complexe, cum ar fi persoane, produse, sau date specifice unei aplicații.

Numele structurii nu este un nume de variabilă, ci un nume de tip de date. **Instrucţiunile de definire a structurilor se vor scrie la începutul fişierului în afara oricărei funcţii.**

Tipuri de Struct in C

Structurile pot fi de două tipuri:

Structuri fără nume: Aceste structuri nu sunt asociate cu un nume specific. Ele sunt folosite atunci când este necesară declararea unei **variabile globale/instanțe** de tip structură.

Structuri cu nume: Aceste structuri sunt asociate cu un nume specific. Ele pot fi folosite în program pentru a declara variabile și pentru a accesa membrii structurii.

Struct in C: structuri cu nume

O structură este declarată folosind cuvântul cheie *struct* urmat de *numele structurii* și de *o listă de membri,* numite câmpurile structurii. Fiecare membru este declarat folosind un tip de date și un nume.

```
struct nume_structura {
    tip_de_date membru1;
    tip_de_date membru2;
    ...
};
```

Struct in C: Exemplu de declarare

De exemplu, următoarea declarație definește o structură pentru a reprezenta un student:

```
struct student {
  char numePrenume[20];
  int varsta;
  float notaMedie;
};
```

Definirea unei structuri nu ocupă spațiu de memorie, ci doar creează un nou tip de structură.

Struct in C: Accesarea membrilor unei structuri

Membrii unei structuri pot fi accesați folosind operatorul de acces *punct* . De exemplu, pentru a accesa membrii **numePrenume, varsta, notaMedie** al structurii student, se poate folosi următoarele expresii:

```
struct student {
  char numePrenume[20];
  int varsta;
  float notaMedie;
};
```

```
int main(){
        struct student s;
        strcpy(s.numePrenume, "Maria");
        s.varsta = 19;
        s.notaMedie = 8.5;
    return 0;
}
```

Struct in C: Inițializare

Structurile pot **fi inițializate în momentul declarării variabilelor** de tip structură:

```
struct student {
  char numePrenume[20];
  int varsta;
  float notaMedie;
};
```

```
int main(){
    struct student s = {"Maria", 19, 9.0};
    printf("%s -> %d ani -> nota %.2f", s.numePrenume, s.varsta, s.notaMedie);
return 0; }
```

Struct in C: Exemplu de program

```
#include <stdio.h>
#include <string.h>
struct student {
  char numePrenume[25];
  int varsta;
  float notaMedie;
};
```

```
int main(){
    struct student s;
    printf("Numele: "); //"%[^\n]%*c"-Enter
    scanf ("%[^\n]%*c", s.numePrenume);
    //scanf ("%s", s.numePrenume);
    printf("Varsta: ");
    scanf("%d", &s.varsta);
    printf("Nota Medie: ");
    scanf("%f", &s.notaMedie);
    printf("\n%s:\nVarsta: %d;\nNota Medie: %.2f.",
s.numePrenume, s.varsta, s.notaMedie);
     return 0;}
```

Struct in C: structuri fără nume

Structurile fără nume sunt definite fără a li se atribui un nume specific. Ele sunt adesea folosite atunci când nu este nevoie să se creeze variabile noi ale structurii în program și când structura este utilizată doar pentru o declarație sau pentru a încapsula date.

```
struct {
    tip_de_date membru1;
    tip_de_date membru2;
    ...
} [one or more structure variables];
```

Struct in C: Exemplu de declarare

De exemplu, următoarea declarație definește o structură pentru a reprezenta 2 studenți:

```
struct {
  char numePrenume[20];
  int varsta;
  float notaMedie;
} student1, student2 ;
```

Definirea acestei structuri **ocupă** spațiu de **memorie** pentru 2 variabile declarate de tipul structurii respective (**2 variabile a câte 28 octeți** fiecare (1B*20+4B+4B – **suma dimensiunilor de memorie ocupată de fiecare componentă/membru/câmp**)).

Struct in C: Exemplu de declarare

De exemplu, următoarea declarație definește o structură pentru a reprezenta 2 studenți:

```
struct {
  char numePrenume[20];
  int varsta;
  float notaMedie;
} student1, student2 ;
```

Ulterior, în program, nu se mai pot declara variabile de tipul structurii date.

Struct in C: Exemplu de program

```
#include <stdio.h>
#include <string.h>
struct {
   char numePrenume[20];
   int varsta;
   float notaMedie;
} s;
int main(){
        strcpy(s.numePrenume, "Gutu Maria");
        s.varsta = 19;
        s.notaMedie = 9.5;
   printf("%s -> %d ani -> nota %.2f", s.numePrenume, s.varsta, s.notaMedie);
   return 0;
}
```

Struct in C: Inițializare structuri fără nume

Structurile pot **fi inițializate în momentul declarării variabilelor** de tip structură:

```
struct student {
  char numePrenume[20];
  int varsta;
  float notaMedie;
} s = {"Maria", 19, 9.0};
```

```
int main(){
    printf("%s -> %d ani -> nota %.2f", s.numePrenume, s.varsta, s.notaMedie);
return 0; }
```

ATENTIE!

Variabila declarată de un tip de structură cu/fără nume, nu poate lua valori astfel: s = {"Maria", 19, 9.0};

Această formă de inițializare/atribuire este acceptată doar în momentul declarării variabilelor de tipul structurii respective.

```
struct student {
  char numePrenume[25];
  int varsta;
  float notaMedie;
};
int main(){
    struct student s = {"Maria", 19, 9.0};
return 0;}
```

```
struct student {
  char numePrenume[25];
  int varsta;
  float notaMedie;
} s = {"Maria", 19, 9.0};
int main(){
    //codul programului
return 0;}
```

Inițializarea poate fi făcută și fiecărui câmp separat:

```
struct student s = {.numePrenume = „Gutu Maria", .varsta = 20, .notaMedie = 9.7568};
```

Struct in C: Încapsularea structurilor

Structurile pot fi și încapsulate în alte structuri, permițând astfel crearea de structuri mai complexe și organizate: **structuri imbricate**.

```
struct Adresa {
    char strada[50];
    int cod_postal;
};
struct Student {
    char numePrenume[50];
    float notaMedie;
    struct Adresa adresa;
};
```

```
struct Student {
    char numePrenume[50];
    float notaMedie;
    struct {
      char strada[50];
      int cod_postal;
    } adresa;
};
```

Structură fără nume

Struct in C: Încapsularea structurilor

```c
struct Adresa {
    char strada[50];
    int cod_postal;
};
```

```c
struct Student {
    char numePrenume[50];
    float notaMedie;
    struct Adresa adresa;
};
```

```c
int main(){
    struct Student s = {"Gutu Maria", 9.0, "Studentilor", 2100};
    printf("%s -> nota: %.2f -> str. %s -> codPostal: %i",
s.numePrenume, s.notaMedie, s.adresa.strada,
s.adresa.cod_postal);
    return 0;
}
```

Struct in C: declarare prin typedef

O structură se poate defini astfel:

```
typedef struct nume_structura {
  declaratii_de_variabile
} [nume nou];
```

Struct in C: Exemplu de declarare

```
typedef struct student {
  char numePrenume[20];
  int varsta;
  float notaMedie;
} Student;
```

typedef este utilizat pentru a crea **un alias/o etichetă** (un nou nume) pentru tipul de date **struct student**. În acest caz, aliasul este **Student**. După această declarație, pentru a declara variabile de tipul acestei structuri se poate folosi direct **Student**: **Student s;**.
Cuvântul cheie **typedef** simplifică sintaxa și face codul mai ușor de citit, deoarece elimină necesitatea de a scrie repetitiv **struct** înaintea numelui structurii în timpul declarării variabilelor.

Struct in C: declarare prin typedef

În cazul utilizării **typedef** în limbajul C pentru o structură, este creat un alias pentru tipul de date structură. Aceasta facilitează folosirea și citirea codului, deoarece se poate de utilizat numele aliasului în locul numelui complet al structurii. În esență, utilizarea **typedef** în acest context nu schimbă structura în sine, ci doar creează un alias pentru tipul de date, facilitând utilizarea acestuia.

Un alias este un nume alternativ a structurii date, este un tip de date, nicidecum o variabilă de tipul structurii respective.

Struct in C: declarare prin 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;
}
```

Struct in C: Exemplu

```
#include <stdio.h>
#include <string.h>
typedef struct student {
  char numePrenume[25];
  int varsta;
  double notaMedie;
} Stud;
int main(){
      Stud s; // struct student s;
      strcpy(s.numePrenume, "Gutu Maria");
      s.varsta = 19;
      s.notaMedie = 9.5;
  printf("%s->%d ani->nota%.2f", s.numePrenume, s.varsta, s.notaMedie);
  return 0; }
```

ATENTIE!

```
typedef struct {
        int data;
        int X;
} S1;
```

```
struct S2 {
        int data;
        int X;
};
```

```
struct {
        int data;
        int X;
} S3;
```

Atât S1 cât și struct S2 formează un tip de date de tip structură.

S3 este o variabilă de tip structură.

Struct in C: Sunt corecte următoarele declarații de structură?

???

Struct in C: Operații permise

- Asignarea unei variabile structură la o altă variabilă structură de același tip
- Obținerea adresei unei variabile structură (&)
- Accesul la componentele (membrii) unei structuri
  - Calificare (operatorul .)
  - Utilizând dereferențierea pointerilor și calificare (operatorul ->)
- Utilizarea operatorului sizeof pentru determinarea dimensiunii unei structuri
- Inițializarea componentelor unei variabile structură este similară cu inițializarea ta blourilor
  - Membrii variabilelor globale sunt inițializați cu zero
  - Membrii variabilelor locale automate sunt neinițializați
- În C, ca și orice alt argument, trimiterea structurilor ca și argumente la apelul func țiilor se face prin valoare.

Struct in C

Atribuirile de structuri în C 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**.

Implementare fără pointeri

Exemplu fără pointeri

```
#include <stdio.h>
#include <string.h>
typedef struct student{
    char nume[20];
    float nota;
} Stud;
```

```
void read(int n, Stud s[10]);
void write(int n, Stud s[10]);
void bubbleInt(int n, Stud s[10]);
void bubbleChar(int n, Stud s[10]);
void clearBuffer();
```

Exemplu

```
int main() {
    int n;
    Stud st[10];
    printf("n="); scanf("%d", &n);
    clearBuffer();     read(n, st);
    printf("--------afisare date nesortate----------\n");
    write(n, st);  bubbleInt(n, st);
    printf("--------afisare date dupa sortare bubbleInt----------\n");
    write(n, st);  bubbleChar(n, st);
    printf("--------afisare date dupa sortare bubbleChar----------\n");
    write(n, st);
    return 0; }
```

Exemplu

```
void read(int n, Stud s[10]){
    for(int i = 0; i < n; ++i){
        printf("Nume: "); scanf("%s", s[i].nume);
        printf("Nota: "); scanf("%f", &s[i].nota);
        clearBuffer();  //fflush(stdin);
    }
}
```

```
void write(int n, Stud s[10]){
    for(int i = 0; i < n; ++i){
        printf("%s: ", s[i].nume);
        printf("%.2f\n", s[i].nota);
    }
}
```

```
void clearBuffer(){
    char c;
    do {
        c = getchar();
    } while(c != '\n');
}
```

Exemplu

```
void bubbleInt(int n, Stud s[10]){
    int i, schimbat;    Stud aux;
    do {
        schimbat = 0;
        for(i = 0; i < n-1; i++)
            if(s[i].nota > s[i+1].nota) {
                aux = s[i];
                s[i] = s[i+1];
                s[i+1] = aux;
                schimbat = 1;
            }
    }while(schimbat);
}
```

```
void bubbleChar(int n, Stud s[10]){
    int i,schimbat;  Stud aux;
    do {
        schimbat = 0;
        for(i = 0; i < n-1; i++)
            if(strcmp(s[i].nume, s[i+1].nume)>0) {
                aux = s[i];
                s[i] = s[i+1];
                s[i+1] = aux;
                schimbat = 1;
            }
    }while(schimbat);
}
```

Implementare cu pointeri

Exemplu cu pointeri

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 şi mai des folosită: */
stud->medie = 9.31;
```

Exemplu cu pointeri

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
typedef struct student {
  char numePrenume[25];
  float notaMedie;
} Student;
```

```
void readData(Student *st, int n);
void writeData(Student *st, int n);
void bubbleSort(Student* std, int n);
void swap(Student *x, Student *y);
void clearBuffer();
Student* allocateMemory(int n);
```

Exemplu

```
int main() {
    Student *std = NULL;
    int n;  printf("n="); scanf("%i", &n);  clearBuffer();
    // std = (Student *) calloc(n, sizeof(Student));
    std = allocateMemory(n);
    readData(std, n);
    printf("\n---afisare date---\n"); writeData(std, n);
    bubbleSort(std, n);
    printf("---afisare date dupa sortare---\n");
    writeData(std, n);
    free(std);
  return 0;}
```

Exemplu cu pointeri

```
Student* allocateMemory(int n){
    Student *std = (Student *) calloc(n, sizeof(Student));
    return std;
}
```

```
void writeData(Student *st1, int n){
for(int i = 0; i < n; ++i){
printf("%s:%g\n",(st1 + i)->numePrenume,(st1 + i)->notaMedie);
    }
}
```

Exemplu cu pointeri

```
void readData(Student *st1, int n){
    for(int i = 0; i < n; ++i){
        printf("Nume & Prenume: ");
        scanf("%[^\n]%*c", (st1 + i)->numePrenume);
        printf("Nota medie: ");
        scanf("%f", &(st1 + i)->notaMedie);
        clearBuffer();
    }
}
```

Exemplu cu pointeri

```
void bubbleSort(Student* std, int n){
    bool flag=true;  Student temp;
    printf("\n\nLista studentilor alfabetic dupa nume\n");
    while(flag){
      flag=false;
      for(int i=0;i<n-1;i++)
       if(strcmp(std[i].numePrenume, std[i+1].numePrenume)>0){
          temp = std[i];     // temp = *(std + i);
          std[i]=std[i+1];   // *(std + i) = *(std + i + 1);
          std[i+1]=temp;     // *(std + i + 1) = temp;
          flag=true;
        }
      }}
```

Aplicații Practice!!!

(maria.gutu@iis.utm.md)