Structuri în limbajul de programare C

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 acelaşi tip). Numele structurii nu este un nume de variabilă, ci un nume de tip de date care se mai numeşte şi eticheta structurii. Instrucţiunile de definire a structurilor se vor plasa la începutul fişierului în afara oricărei funcţii.

O structură se poate defini astfel:

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

Exemplu de ***declarare de structură***:

```
struct student {
  char numePrenume[10];
  int varsta;
  float notaMedie;
};
```

Variabilele declarate în interiorul structurii se numesc "câmpurile" structurii: ***numePrenume[10], varsta*** și ***notaMedie***, iar ***student*** este eticheta structurii, adică numele structurii și a tipului de date definit și stabilit de programator.

Exemplu de program cu tip de date structură:

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

struct student {
  char numePrenume[25];
  int varsta;
  float notaMedie;
};

int main()
{
    struct student s;
    // "%[^\n]%*c"  se va citi caracterele introduse de la tastatura
    //  pana cand nu se va apasa tasta Enter
    printf("Numele: ");
    scanf ("%[^\n]%*c", s.numePrenume);
    // "%s" se va citi caracterele pana la primul caracter alb (spatiu sau Enter)
    //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;
}
```

O altă alternativă ar fi:

```
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;
}
```

***Exemplu de program** cu tip de date structură rezultat din typedef:*

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

typedef struct student {
  char numePrenume[25];
  int varsta;
  double notaMedie;
} Stud;

int main()
{
    Stud s;

    printf("Numele: ");
    scanf ("%[^\n]%*c", s.numePrenume);

    printf("Varsta: ");
    scanf("%d", &s.varsta);
    printf("Nota Medie: ");
    scanf("%lf", &s.notaMedie);

    printf("\n%s:\nVarsta: %d;\nNota Medie: %.2lf.", s.numePrenume, s.varsta, s.notaMedie);

    return 0;
}
```

Declararea variabilelor de tip structură

Forma generală:

```
struct nume_structura {
 declaratii_de_variabile
} [one or more structure variables];
```

Exemplu de declarare de structură:

```
struct student {
  char numePrenume[10];
  int varsta;
  float notaMedie;
} s1, s2;
```

Exemplu de program cu tip de date structură:

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

struct student {
  char numePrenume[25];
  int varsta;
  float notaMedie;
} s1;

int main()
{
    printf("Numele: ");
    scanf ("%[^\n]%*c", s1.numePrenume);

    printf("Varsta: ");
    scanf("%d", &s1.varsta);
    printf("Nota Medie: ");
    scanf("%f", &s1.notaMedie);

    printf("\n%s:\nVarsta: %d;\nNota Medie: %.2f.", s1.numePrenume, s1.varsta, s1.notaMedie);
    return 0;
}
```

Iniţializarea unor variabile de tip structură:

```
struct student s1 = {"Carp Ion", 30, 8.85};
```

sau, inițializarea fiecărui câmp separat:

```
struct student s1 = {.numePrenume = "Ursu Ana", .varsta = 25, .notaMedie = 9.7568};
```

Exemplu de program:

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

struct student {
  char numePrenume[25];
  int varsta;
```

```
  float notaMedie;
};

int main()
{
    struct student s = {"Carp Ion", 30, 8.85};
    printf("\n%s:\nVarsta: %d;\nNota Medie: %.2f.\n", s.numePrenume, s.varsta, s.notaMedie);

    struct student s1 = {.numePrenume = "Ursu Ana", .varsta = 25, .notaMedie = 9.7568};
    printf("\n%s:\nVarsta: %d;\nNota Medie: %.2f.", s1.numePrenume, s1.varsta, s1.notaMedie);

    return 0;
}
```

Alt exemplu de utilizare: După cum se vede mai jos trebuie facută diferența când definim un tip şi când declarăm o variabilă 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;
}
```

Manipularea datelor unui **tabloul unidimensional** cu elemente de tip **structură fără pointeri**:

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

void read(int n);
void write(int n);
void bubbleInt(int n);
void bubbleChar(int n);
void clearBuffer();

struct student st[10];
struct student aux;

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

```
void read(int n){
    for(int i = 0; i < n; ++i){
        printf("Nume: "); scanf("%[^\n]%*c", st[i].nume);
        printf("Nota: "); scanf("%f", &st[i].nota);
        clearBuffer();  //fflush(stdin);
    }
}
void write(int n){
    for(int i = 0; i < n; ++i){
        printf("%s: ", st[i].nume);
        printf("%.2f\n", st[i].nota);
    }
}

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

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

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.

```
//Returnarea unei structuri dintr-o funcție
#include <stdio.h>
struct student
{
    char nume[20];
    int varsta;
};
// prototipul functiilor
struct student input();
void output( struct student s1);

int main()
{
    struct student s;
    s = input();
    output(s);
    return 0;
}
struct student input()
{
  struct student s1;
  printf("Nume: ");
  scanf ("%[^\n]%*c", s1.nume);
  printf("Varsta: ");
  scanf("%d", &s1.varsta);
  return s1;
}
void output( struct student s1){
    printf("\nAfisare date:\n");
    printf("Nume: %s", s1.nume);
    printf("\nVarsta: %d", s1.varsta);
}
```

```
//Passing struct by reference
#include <stdio.h>
typedef struct subject
{
    float notaSDA, notaPC;
} Subject;
```

```
void suma(Subject st1, Subject st2, Subject *avgSubject);

int main() {
    Subject st1, st2, avgSubject;

    printf("st1.notaSDA = ");
    scanf("%f", &st1.notaSDA);
    printf("strc1.notaPC = ");
    scanf("%f", &st1.notaPC);

    printf("strc2.notaSDA = ");
    scanf("%f", &st2.notaSDA);
    printf("strc2.notaPC = ");
    scanf("%f", &st2.notaPC);

    suma(st1, st2, &avgSubject);
    printf("\navgSubject.SDA = %.1f", avgSubject.notaSDA);
    printf("\navgSubject.PC = %.1f", avgSubject.notaPC);

    return 0;
}
void suma(Subject st1, Subject st2, Subject *avgSubject)
{
     avgSubject->notaSDA = (st1.notaSDA + st2.notaSDA)/2;
     avgSubject->notaPC = (st1.notaPC + st2.notaPC)/2;
}
```

```
//Dynamic memory allocation of structs
#include <stdio.h>
#include <stdlib.h>
struct student {
   char nume[30];
   int age;
   float media;
};
int main() {
   struct student *ptr;
   int n;
   printf("Nr studenti: ");
   scanf("%d", &n);
   // allocating memory for n numbers of struct student
   ptr = (struct student*) malloc(n * sizeof(struct student));
   for(int i = 0; i < n; ++i)
   {
       printf("Nume: "); scanf("%s", (ptr+i)->nume);
       printf("Varsta: "); scanf("%i", &(ptr+i)->age);
       printf("Media: "); scanf("%f", &(ptr+i)->media);
   }
   printf("Afisarea Informatiei:\n");
```

```
   for(int i = 0; i < n; ++i)
    printf("Nume: %s\tVarsta: %d\tMedia: %.2f\n", (ptr+i)->nume, (ptr+i)->age, (ptr+i)->media);
   return 0;
}
```

```
#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);

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;
}
Student* allocateMemory(int n){
    Student *std = (Student *) calloc(n, sizeof(Student));
    return std;
}
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();
    }
}
void writeData(Student *st1, int n){
    for(int i = 0; i < n; ++i){
```

```
        printf("%s: %g\n", (st1 + i)->numePrenume, (st1 + i)->notaMedie);
    }
}
/*void bubbleSort(Student* std, int n){
    bool flag=true;
    Student temp;
    printf("\n\nLista studentilor in ordinea alfabetica a numelui\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;
            }
        }
    }
}*/
void swap(Student *x, Student *y){
    Student temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
void bubbleSort(Student* std, int n){
    bool flag=true;
    printf("\n\nLista studentilor in ordinea alfabetica a numelui\n");
    while(flag){
        flag=false;
        for(int i=0;i<n-1;i++){
            if(strcmp(std[i].numePrenume, std[i+1].numePrenume)>0){
               // swap(&std[i], &std[i+1]);
               swap((std+i), (std+i+1));
                flag=true;
            }
        }
    }
}
void clearBuffer() {
        char c;
        do {
                c = getchar();
        } while (c != '\n' && c != EOF);
}
```