# TIPURILE DE DATE STRUCTURĂ, UNIUNE ŞI ENUMERARE

## 1. Conţinutul lucrării

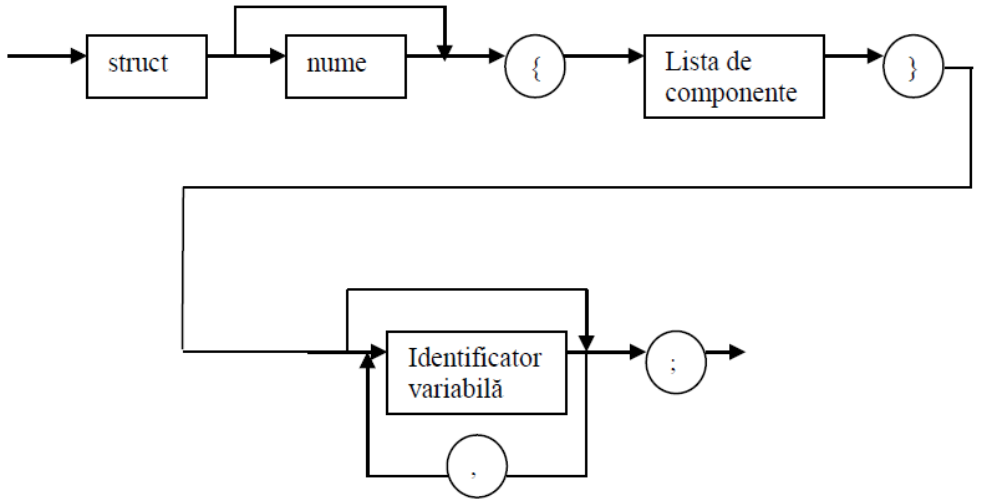
În lucrare sunt prezentate tipurile definite de utilizator structură, uniune şi enumerare, accesul la componentele lor şi asignarea de nume pentru aceste tipuri.

## 2. Consideraţii teoretice

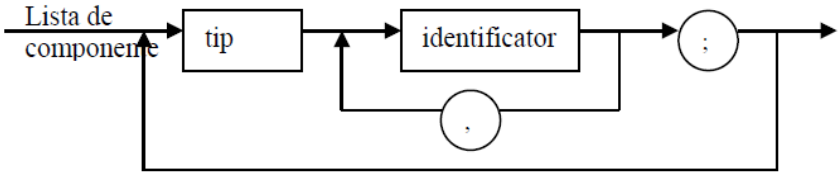
### 2.1. Tipul de date "structură"

O **structură** conţine mai multe componente de tipuri diferite (predefinite sau definite de utilizator), grupate conform unei ierarhii.

Declaraţia unei structuri se poate face astfel:

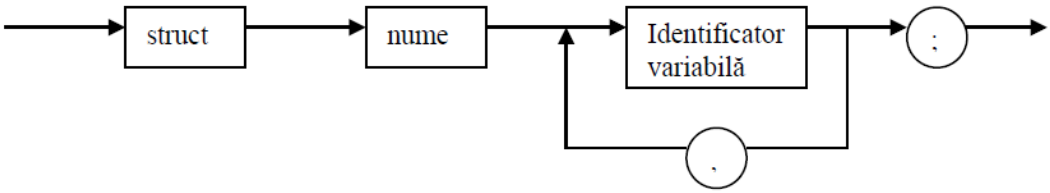



Lista de componente (câmpuri) a unei structuri se defineşte astfel:




Observaţie: În declaraţia unei structuri nu pot lipsi simultan numele structurii şi identificatorii de variabile.

O variabilă structură de tipul "nume" poate fi declarată şi ulterior conform diagramei de mai jos, cu menţiunea că cuvântul cheie "struct" poate lipsi în limbajul C++:



Următoarele trei exemple sunt echivalente pentru declararea variabilelor *stofa*, *hartie*, *motor*:

```
a) struct material {
                        int cod;
                        char den [30];
                        char um [10];
                        float cantitate;
                        float pret_unit;
                    } stofa, hartie, motor;

b) struct material{
                        int cod;
                        char den [30];
                        char um [10];
                        float cantitate;
                        float preţ_unitar;
                    };
   struct material stofa, hartie, motor;
        sau
   material stofa, hartie, motor;

c) struct {
              int cod;
              char den [30]
              char um [10];
              float cantitate;
              float pret_unitar;
          } stofa, hartie, motor;
```

Accesul la componentele (câmpurile) unei structuri se poate face prin procedeul de calificare:
**identificator_variabilă.identificator_câmp;**

Exemple: stofa.den
hartie.cantitate

Procedeul de calificare pătrunde din aproape în aproape în ierarhia structură.

În cazul în care se transmite adresa unei variabile de tip structură (pointer la o structură) ca parametru la apelul unei funcții, accesul la câmpurile structurii se poate face în două moduri. De exemplu:

**void f(struct material *p, ...);**

Apelul se va face prin:

**f(&stofa, ...)**

În funcţie, selectarea unui câmp se face astfel:

(*p).den
(*p).cantitate

sau înlocuind **(*p).** prin **p->** , ca mai jos:

```
p->den
p->cantitate
```

O structură de acelaşi tip se poate atribui direct una alteia, moment în care se copiază valorile tuturor câmpurilor:

```
material alfa, beta;
alfa=beta;
```

### 2.2. Tipul de date "uniune"

În momente diferite ale execuţiei, se pot păstra în aceeaşi zonă de memorie date de tipuri diferite pentru economisirea memoriei sau din motive ale concepţiei programului. Acest lucru se face grupând toate datele care se alocă în aceeaşi zonă de memorie. Structura utilizator obţinută se numeşte **uniune.** Sintaxa uniunii este identică cu cea a structurii, singura deosebire constând în înlocuirea cuvântului cheie "**struct**" cu "**union**" în diagramele de sintaxă de la punctul 2.1.

De menţionat că zona de memorie rezervată are dimensiunea componentei care necesită cea mai multă memorie pentru reprezentare.

Accesul la componente se face identic ca la structură.

De menţionat că programatorul trebuie să cunoască în fiecare moment care dată este reprezentată.

Exemplu:

```
union alfa {
                char c[5];      /* reprezentare pe 5 octeţi */
                short i;        /* reprezentare pe 2 octeţi */
                long j;         /* reprezentare pe 4 octeţi */
            };
union alfa x;
strcpy(x.c, "ABCD");
```

Variabila x are reprezentarea în hexazecimal, astfel:

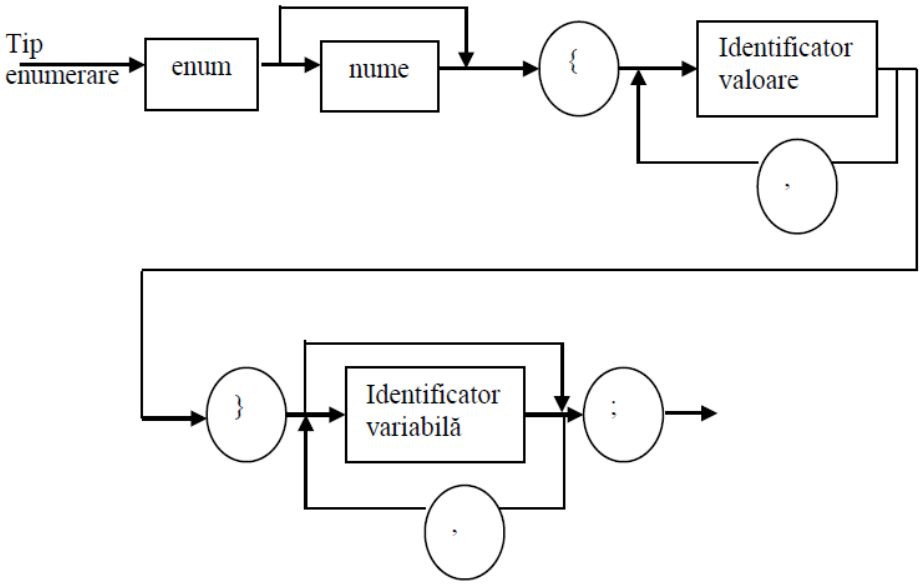
| | | | | |
|---|---|---|---|---|
| 41 | 42 | 43 | 44 | 00 |

Dacă se accesează componenta x.i, atunci aceasta va avea 4241 în hexazecimal (datorită reprezentării *Little-endian*), adică 16961 în zecimal.

În schimb, aceeaşi zonă de memorie interpretată ca x.j (long) va avea valoarea 44434241 în hexazecimal (datorită reprezentării *Little-endian*), adică 1145258561 în zecimal.

### 2.3. Tipul de date "enumerare"

Tipul **enumerare** permite programatorului să folosească nume sugestive pentru valori numerice. Diagrama de sintaxă pentru tipul enumerare este asemănătoare cu tipurile structură şi uniune. Deosebirea constă în faptul că lista de componente este formată numai din identificatori de valoare întreagă 0, 1, 2, ... :




Următoarele trei exemple sunt echivalente pentru declararea variabilei *sapt_vacanta*:

a) enum sapt {luni, marti, miercuri, joi, vineri, sambata, duminica};
   enum sapt sapt_vacanta;

b) enum sapt {luni, marti, miercuri, joi, vineri, sambata, duminica} sapt_vacanta;

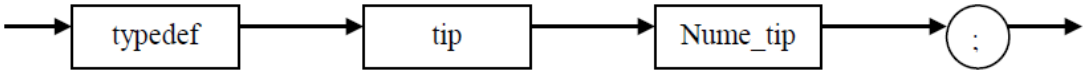
c) enum {luni, marti, miercuri, joi, vineri, sambata, duminica} sapt_vacanta;

Atribuire posibilă:
sapt_vacanta=vineri;

Identificatorii luni, marti, … , au valorile 0, 1, ….

### 2.4. Declararea tipurilor de date prin nume simbolice

În limbajul C/C++ se poate atribui un nume simbolic unui tip predefinit sau unui tip utilizator. Diagrama de sintaxă pentru asignarea unui nume simbolic "nume_tip" unui "tip" predefinit sau utilizator este următoarea:




Exemple:

a) typedef struct {
                int i;
                float j;
                double x;
            } ALFA;
   ALFA y, z;

b) typedef struct {
float re;
float im;
} COMPLEX;
COMPLEX x, y;

c) typedef union {
char x[10];
long cod;
} BETA;
BETA u, v;

d) typedef enum {false, true} BOOLEAN;
BOOLEAN k, l;

### 2.5. Exemple de programe

Programul următor prezintă operaţii asupra numerelor complexe, folosind tipul "structură".

```
/*Programul L10Ex1 */

#include <stdio.h>
#include <stdlib.h>
typedef struct
        {
          float re, im;
        } COMPLEX;

void aduna(COMPLEX *a, COMPLEX *b, COMPLEX *c)
 /* transmiterea parametrilor prin pointeri */
{
   c->re=a->re+b->re;
   c->im=a->im+b->im;
}

void scade(COMPLEX a, COMPLEX b, COMPLEX *c)
 /* transmiterea parametrilor de intrare prin valoare si a rezultatului prin pointer */
{
   c->re=a.re-b.re;
   c->im=a.im-b.im;
}

void produs(COMPLEX *a, COMPLEX *b, COMPLEX *c)
/*transmiterea parametrilor prin pointeri */
{
   c->re=a->re*b->re-a->im*b->im;
   c->im=a->im*b->re+a->re*b->im;
}
```

```
void impartire(COMPLEX *a, COMPLEX *b, COMPLEX *c)
/*transmiterea parametrilor prin pointeri */
{
   float x;
   x=b->re*b->re+b->im*b->im;
   if (x==0) {
        printf("\nImpartire la zero!\n");
        exit(1);
  }
  else{
         c->re=(a->re*b->re+a->im*b->im)/x;
         c->im=(a->im*b->re-a->re*b->im)/x;
  }
}

 int main()
 /* Operaţii asupra numerelor complexe */
 {
   COMPLEX a,b,c;
   char ch;
   ch='D';
   while ((ch=='D')|| (ch=='d'))
   {
     printf("\nIntroduceti primul numar complex\n");
     printf("a.re=");scanf("%f",&a.re);
     printf("a.im=");scanf("%f",&a.im);
     printf("\nIntroduceti al doilea numar complex\n");
     printf("b.re=");scanf("%f",&b.re);
     printf("b.im=");scanf("%f",&b.im);

     aduna(&a,&b,&c);
     printf("\n(%f+j*%f)+(%f+j*%f)=%f+j*%f\n", a.re,a.im,b.re,b.im,c.re,c.im);

     scade(a,b,&c);
     printf("\n(%f+j*%f)-(%f+j*%f)=%f+j*%f\n", a.re,a.im,b.re,b.im,c.re,c.im);

     produs(&a,&b,&c);
     printf("\n(%f+j*%f)*(%f+j*%f)=%f+j*%f\n", a.re,a.im,b.re,b.im,c.re,c.im);

     impartire(&a,&b,&c);
     printf("\n(%f+j*%f)/(%f+j*%f)=%f+j*%f\n", a.re,a.im,b.re,b.im,c.re,c.im);

     printf("\nCONTINUATI?DA=D/d,Nu=alt caracter " );
     scanf("%*c%c",&ch);
   }

   return 0;
 }
```

Programul următor prezintă operaţii asupra datelor de tipul "union":

```
/* Programul L10Ex2 */

#include <stdio.h>
#include <string.h>
/* Exemplu de folosire a tipului "union" */
int main()
{
  typedef union{
      char ch[10];
      short x;
      long y;
      float f;
  } alfa;
  alfa  a;
  strcpy(a.ch,"ABCDEFGHI");
  printf("\nDimensiunea zonei de memorie rezervata = %d octeti\n", sizeof(a));
  printf("\nCONTINUTUL ZONEI:\n");
  printf("\n-sir de caractere: %s",a.ch);
  printf("\n-intreg de tipul short: %d(%x in hexa)",a.x,a.x);
  printf("\n-intreg de tipul long: %ld(%lx in hexa)",a.y,a.y);
  printf("\n-real de tipul float: %g",a.f);
  return 0;
}
```

Programul următor prezintă operații asupra datelor de tipul "enum":

```
/* Programul L10Ex3 */

#include <stdio.h>
#include <conio.h>
/* Exemplu de folosire a tipului "enum" */
int main()
{
  typedef enum{zero,unu,doi,trei,patru,cinci} NR;
  NR x,y;
  int z,w;
  x=doi;     /* x=2 */
  y=trei;     /*x=3*/
  z=x+y;
  w=x*y;
  printf("\nz=%d w=%d\n",z,w);
  x=2;y=3;     /* o astfel de atribuire nu este intocmai indicata; ar trebui folosite numele
                  sugestive din cadrul enum-ului*/
  z=x+y;
  w=x*y;
  printf("\nz=%d w=%d\n",z,w);
  return 0;
}
```

## 3. Mersul lucrării

3.1. Folosind tipul structură pentru o dată curentă an, lună, zi, să se scrie un program pentru a afişa a câtea zi din an este ziua respectivă şi câte zile au mai rămas până la sfârşitul anului.

3.2. Folosind tipul structură pentru data voastră de naştere şi ştiind că în anul curent vă aniversaţi ziua de naştere în ziua de x [luni, marţi, ..., duminică], scrieţi un program pentru a afişa ziua (din săptămână) în care v-aţi născut.

3.3. Să se scrie un program modularizat care citeşte datele legate de studenţii unei grupe: nume, data naşterii, adresa şi îi afişează în ordine alfabetică.

3.4. Să se scrie un program pentru calculul valorii unui polinom de gradul n cu coeficienţi complecşi pentru o valoare complexă. Calculul se va face cu ajutorul unei funcţii.

3.5. Să se introducă tipul "raţional" ca o structură formată din numărător şi numitor. Să se scrie funcţii de simplificare, adunare, scădere, înmulţire, împărţire, ridicare la putere.

3.6. Folosind tipul uniune, care conţine câmpurile necesare pentru a putea reprezenta un cerc, un dreptunghi, un pătrat, un triunghi, să se scrie o funcţie pentru a calcula aria figurii respective.

3.7. Folosind tipul enumerare, să se introducă tipul "boolean". Să se scrie o funcţie de ordonare crescătoare a unui tablou de numere reale folosind metoda bulelor şi o variabilă de tipul boolean.

3.8. Se citeşte un şir de caractere format din litere şi cifre. Să se indice frecvenţa de apariţie a caracterelor întâlnite în şir folosind un tablou de elemente structură (câmpurile structurii sunt: caracterul și frecvenţa de apariție).

3.9. Un camion poate fi încărcat cu maxim *m* kilograme. Numele materialelor disponibile, cantitățile exprimate în kilograme și prețul per kilogram sunt cunoscute. Să se scrie un program care determină o posibilă încărcare a camionului cu anumite cantități din materiale astfel încât valoarea totală să fie maximă.

3.10. Un polinom rar este un polinom în care majoritatea monoamelor sunt zero. Scrieți un program care memorează eficient un polinom rar de grad *m* și funcții pentru afișarea lui și găsirea valorii lui într-un anumit punct.

3.11. O matrice rară este o matrice în care majoritatea elementelor sunt zero. Scrieți un program care memorează eficient matrice rare și funcții pentru calculul sumei, diferenței și produsului de perechi de matrice rare.