Facultatea de Inginerie Electrică

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

EUROPEAN UNIVERSITY OF TECHNOLOGY

Facultatea de Inginerie Electrica

PCLP 2

Programarea calculatoarelor si limbaje de programare 2

PCLP2

An I semestrul II

"Coding is easy when you C it in action."

Cap. 5

Alte tipuri de date structurate in C

5. 1. Uniuni

- ❑ Definire. Exemple
- ❑ Accesul la elemente
- ❑ Initializarea elementelor
- ❑ Alocarea memoriei
- ❑ Pointeri la uniuni
- ❑ Tablouri de uniuni
- ❑ Transmiterea uniunilor catre functii

5. 2. Enumerari

5. 3. Tipuri definite de utilizator

5. 4. Campuri de biti

5.1 Uniuni

Uniuni-definire

DEFINIRE

Uniune: variabila care **poate pastra, la momente diferite, obiecte de tipuri si marimi diferite in aceeasi zona de memorie**. Aceasta variabila va ocupa suficienta memorie ca sa poata stoca cel mai mare dintre tipurile din componenta ei.

SINTAXA

Declarare tip uniune:

```
union tip_uniune
    { tip var1;
      tip var2;
      …
     } ;
```

Declarare uniune:

```
union tip_uniune   lista_variabile_uniune;
```

Declarare uniune intr-o singura instructiune:

```
union tip_uniune
      { tip var1;
        tip var2;
        …
       } lista_variabile_uniune;
```

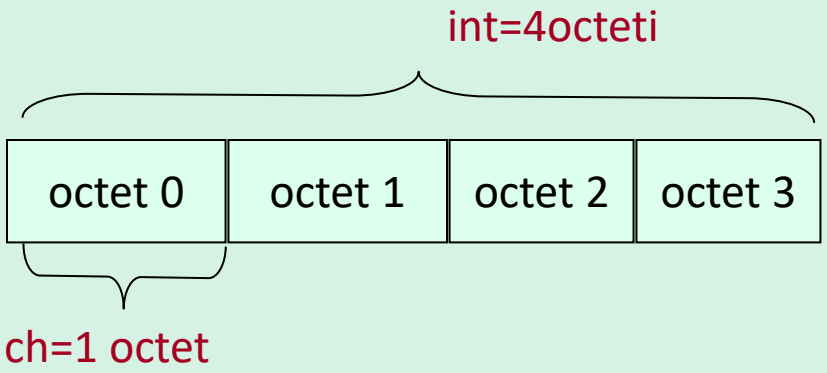
5.1 Uniuni

Uniuni-exemple

EXEMPLE

Ex. : declarare uniune :

```
union tip
        { int i;
          char ch;
        } cont;
```



Modul de utilizare a uniunii cont de catre variabilele i si ch

5.1 Uniuni

Uniuni-accesul la emente

DEFINIRE

Similar cu accesul la elementele unei structuri:

Operatorul punct (.) : nume_uniune.element

Operatorul sageata (->) : pointer_uniune-> element

EXEMPLE

Ex. 1: acces la elementele unei uniuni :

```
union tip
    { int i;
      char ch;
     } cont;
union tip *p; p=&cont;

cont.i=10;
p->i=10;  sau   (*p).i=10
```

5.1 Uniuni

Uniuni-accesul la emente

EXEMPLE

Ex. 2: definire uniune , initializare si afisare termeni

```
#include <stdio.h>
int main(void) {
union { int a, b; } reunin;
reunin.a = 2;
printf("a=%d\n", reunin.a);
reunin.b = 4;
printf("b=%d\n", reunin.b);
printf("a=%d\n", reunin.a);
return 0;
}
```

```
a=2
b=4
a=4
```

5.1 Uniuni

Alocarea memoriei pentru structuri si uniuni

DEFINIRE

Structuri : zona de memorie ocupata >= **suma in octeti** a elementelor (Ex.1).

Uniuni: zona de memorie ocupata =**maximul octeti** ocupati de elemente (Ex.2).

EXEMPLE

Ex.1. : Determinarea marimii zonei de memorie ocupate de structura x

```
struct s
{   char ch;  //1 octet
    int i;    //4 octeti
    float f; //4 octeti
} x;

// sizeof (x) =1+4+4 =9
```

Ex.2. : Determinarea marimii zonei de memorie ocupate de uniunea y

```
union u
  {   char ch;  //1 octet
       int i; //4 octeti
       float f; //4 octeti
  } y;

// sizeof (y) =max(1,4,4)=4
```

5.1 Uniuni

Alocarea memoriei pentru structuri si uniuni

EXEMPLE

Ex. 1: definire uniuni si afisare dimensiune in octeti

```
#include <stdio.h>
int main()
{union test1        {int a,b;} t;
 union test2        {int x;char y;} r;
 union test3        {int arr[10];char z[10];} q;
printf ("sizeof(test1) = %d, sizeof(test2) = %d,sizeof(test3) = %d", sizeof(t), sizeof(r), sizeof(q));
return 0;}
```

Ce se afiseaza in urma executiei?

```
sizeof(test1) = 4, sizeof(test2) = 4,sizeof(test3) = 40
```

5.1 Uniuni

Alocarea memoriei pentru structuri si uniuni

EXEMPLE

Ex. 2: definire tablouri uniuni si afisare dimensiune in octeti

```
#include <stdio.h>
int main()
{union test1    {int a,b;} t[10];
union test2     {int x;char y;} r[10];
union test3     {int arr[10];char z[10];} q[10];
printf ("sizeof(test1) = %d, sizeof(test2) = %d,sizeof(test3) = %d", sizeof(t), sizeof(r), sizeof(q));
return 0;}
```

Ce se afiseaza in urma executiei?

```
sizeof(test1) = 40, sizeof(test2) = 40,sizeof(test3) = 400
```

5.1 Uniuni

Alocarea memoriei pentru structuri si uniuni

EXEMPLE

Ex. 3: definire structura si uniune imbricate

```
#include <stdio.h>
 int main()
{struct tip
    {char name[10];
     union
        { char sval[20]; int cod;
        } u;
 } tab[10];
printf ("%d", sizeof(tab));
return 0;}
```

Ce se afiseaza in urma executiei?

```
300
```

5.1 Uniuni

Alocarea dinamica a memoriei pentru uniuni

EXEMPLE

Ex. : alocare dinamica

```
union unitate
        {    char denumire[60], caracteristici[100];
             int cantitate, garantie;
             double pret;
       };
         union unitate *p;
         p=(union unitate*)malloc(20*sizeof(union unitate));
```

TEST

1 . Se considera urmatoarea secventa de instructiuni:

```
struct unitate
{       char denumire[60], caracteristici[100];
        int cantitate, garantie;
        double pret;    };
struct unitate *p;
p=(struct unitate*)malloc(20*sizeof(struct unitate));
```

Cat spatiu de memorie se aloca dinamic prin pointerul p, daca tipul char se reprezinta pe 1 octet, int si float pe 4 octeti, iar double pe 8 octeti (2p)?

a)176 b) 3520 c) 1440 d) 3440

Raspuns correct

b)

TEST

2.Se considera urmatoarea secventa de instructiuni:

```
union unitate
{        char denumire[60], caracteristici[100];
         int cantitate, garantie;
         double pret;      };
union unitate *p;
p=(union unitate*)malloc(20*sizeof(union unitate));
```

Cat spatiu de memorie se aloca prin pointerul p, daca tipul char se reprezinta pe 1 octet, int si float pe 4 octeti, iar double pe 8 octeti (2p)?

a)176 b) 3520 c) 100 d) 2000

Raspuns correct

d)

TEST

3. Se considera urmatoarea secventa de instructiuni:

```
union unitate
{       char nume[60], adresa[50];
        double nota[10];  };
union unitate p[10];
```

Cat spatiu de memorie se aloca prin p, daca tipul char se reprezinta pe 1 octet, int si float pe 4 octeti, iar double pe 8 octeti (2p)?

a)600 b) 1900 c) 800 d) 1200

Raspuns correct

c)

5.2 Enumerari

Declarare enumerari

DEFINIRE

Enumerare: set de constante ce specifica toate valorile pe care le poate lua variabila de acel tip.

SINTAXA

Declarare enumerare: similar cu declararea structurilor

enum [nume_enum] { lista enum separate cu virgula} lista_variabile_enum ;
typedef enum [nume_enum] {lista enum} nume_tip;

- nume_ enum este numele noului tip de date creat si este optional
- lista enum este considerata lista de constante de tip intreg, primul element din lista are valoarea 0, al doilea valoarea 1, s.a.m.d. , daca nu se initializeaza cu alte valori
- lista_variabile_enum este lista variabilelor de tipul nume_enum

Efect: se declara variabile de tipul enumerare , acest tip permitand definirea unei liste de contante intregi cu nume in vederea folosirii de nume sugestive pentru valori numerice.

5.2 Enumerari

Exemple

EXEMPLE

Ex.1.: declarare variabila tip enumerare numita bani de tip monede

```
enum monede {dolar,euro,leu,yen,forint } ; //declarare tip  enumerare
 enum monede bani;   //declarare variabila enumerare
```

Ex. 2.: declarare variabila tip enumerare numita logic de tip Boolean

```
enum Boolean {false, true} logic;   //false=0, true=1,
//se poate utiliza in expresii conditionale: logic==false sau logic == true
```

Ex.3.: declarare variabila tip enumerare fara specificarea nume_enum

```
enum { ileg,ian,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna ;
//expresii echivalente: luna =3;  luna=mar; (pentru ca mar=3)  //sau
enum {ian=1,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna;
```

Ex.4.: declarare variabila tip enumerare utilizand operatorul typedef

```
typedef enum { false, true} BOOLEAN;
BOOLEAN logic;
```

5.2 Enumerari

Accesul la elemente

Direct , utilizand numele si nr. de ordine din lista de enumerare

EXEMPLE

Ex. : declarare variabila tip enumerare numita bani de tip monede

```
enum monede
    {dolar,euro,leu,yen,forint } ; //declarare tip  enumerare
enum monede bani;   //declarare variabila enumerare
```

instructiuni permise

```
bani=leu;   //echivalent cu bani=2 pentru ca leu=2
if (bani==forint) printf("Moneda este un forint")
printf("%d, %d", dolar,leu);   // va tipari valorile 0,2
```

5.2 Enumerari

Initializarea elementelor

Implicit elementele din lista enum sunt initializate cu valori pornind de la 0,1,…
Initializarea elementelor cu alte valori decit cele implicite se face utilizind semnul egal urmat de o valoare intreaga , modificandu-se si valorile elementelor ce urmeaza dupa valoarea initializata

EXEMPLE

Ex. : initializare elemente enumerare:

```
enum monede
    {dolar,euro,leu=100,yen,forint } ;
enum monede bani;
printf("%d, %d, %d,%d,%d", dolar,euro,leu,yen,forint);
    // va tipari valorile 0,1,100,101,102
```

- Elementele din lista de enumerare nu sunt siruri de caractere ci sunt o eticheta pentru valori intregi!

5.2 Enumerari

Initializarea elementelor

EXEMPLE

Ex. : initializare elemente enumerare cu aceleasi valori:

```
#include <stdio.h>
enum State { Working = 1, Failed = 0, Freezed = 2 };
int main() {
    printf("%d, %d, %d", Working, Failed, Freezed);
    return 0;
}
```

```
1, 0, 2
```

Ex. : initializare elemente enumerare:

```
#include<stdio.h>
int main()
{ enum status {pass, fail, absent}
      stud1,stud2,stud3;
 stud1 = pass;
 stud2 = absent;
 stud3 = fail;
 printf("%d %d %d\n", stud1, stud2, stud3);
 return 0;}
```

Ce se afiseaza in urma executiei?

```
0 2 1
```

TEST

1 . Considerand secventa de mai jos:

```
enum Culori {gri, alb, albastru, rosu};
struct calculator {
    char Den[10];
    int Garantie;
    enum Culori Culoare;    };
struct calculator PC={"PC HP 2856",3,gri};
struct calculator PC1,*p ;
p=&PC;
```

Indicati care dintre expresiile de mai jos nu este corectă?

a) PC1.Garantie= 2;
b) PC1=PC;
c) p.Garantie = 1; (*p).garantie = 1;
d) p->Culoare = alb;

Raspuns correct

c)

5.3 Tipuri definite de utilizator

Operatorul typedef

DEFINIRE

typedef tip nume_nou
tip = orice tip de date existent
nume_nou = numele nou dat tipului tip

EXEMPLE

Ex.1 : declaratie de tip float

```
typedef float bilant; //bilant este un alt nume pentru tipul float
bilant scadent;       //se declara variabila scadent de tipul bilant adica float
typedef bilant total; // total  este un alt nume pentru tipul bilant adica pentru tipul float;
total balanta;
```

Nu se creeaza de fapt nici un tip nou de date , ci numai un nou nume pentru un tip de date existent.

5.3 Tipuri definite de utilizator

EXEMPLE

Ex. 2: declaratie de tip structura

```
typedef struct data{  int zi;
                      char luna[11];
                      int an;
                    } dc;
dc datan,dataa,*p;
p=&datan;
```

Ex. 3: declaratie de tip uniune

```
typedef union       { char nume[40];
                      int urm;
                      long cod;
                    } zc;
zc sir,*p;
p=&sir;
p->nume[0]='A';
p->cod=17;
```

Ex. 4: declaratie de tip structura

```
typedef struct {  double real;
                  double imag;
                } complex;
complex z,tz[10];
z.real=1.5;
tz[1].real=2.7;
tz[1].imag=-1.;
```

Ex. 5: declaratie de tip pointer

```
typedef int *POINTER;
POINTER p, t[20];
//echivalent cu: int *p,*t[20];
```

Ex. 6: declaratie de tip tablou

```
typedef  double a[100]
a nr;   //nr va fi tablou de maxim 100 de elemente de tip double
```

TEST

Se considera urmatoarea secventa de instructiuni:

```
typedef struct laptop
{       char denumire[20], caracteristici[20];
        int cantitate, garantie;
        double pret;
} unitate;
unitate *p;
p=new unitate[100];
```

Cat spatiu de memorie se aloca dinamic prin pointerul p, daca tipul char se reprezinta pe 1 octet, int si float pe 4 octeti, iar double pe 8 octeti ?

a) 4600 b) 200 c) 560 d) 5600

Raspuns correct

d)

5.4 Campuri de biti

DEFINIRE

Camp de biti: este un element al unei structuri care cuprinde unul sau mai multi biti adiacenti.
Efect: se pot accesa prin nume, unul sau mai multi biti dint-un octet sau cuvant. Campurile de biti se pot grupa formand o structura .

SINTAXA

Format declarare:

```
struct nume_struct {
        tip nume1: lungime;
        tip nume2: lungime;
        …
        tip numeN: lungime;
        } lista_variabile;
```

unde tip = tipul campului de biti ce poate fi int, unsigned sau signed.
lungime = nr. de biti dintr-un camp

5.4 Campuri de biti

- Campul de biti permite accesul la un singur bit
- Campul de biti cu lungimea 1 trebuie declarat de tip unsigned (pentru ca un singur bit nu poate avea semn).
- Campurile de biti sunt utilizate frecvent pentru analiza intrarii echipamentelor hardware

Restrictii de utilizare a variabilelor de tip camp de biti

- Nu se poate obtine adresa unui camp de biti cu operatorul &
- Nu pot fi utilizate intr-un tablou

5.4 Campuri de biti

EXEMPLU

Ex.1 : camp de biti utilizat in cadrul unei structuri

```
struct angajat {
    struct adr adrese ;
    float salar ;
    unsigned activ:    1        //statut angajat: activ sau intrerupt
    unsigned orar:     1        //plata orara sau lunara
    unsigned impozit:1          //impozit rezultat
};
```

Se defineste o inregistrare intr-o baza de date despre un angajat care foloseste numai un octet pentru a pastra 3 informatii:

- statutul angajatului,
- daca e platit la ora sau cu salar lunar
- modul in care se calculeaza impozitul (prestari servicii/carte de munca)

5.4 Campuri de biti

SINTAXA

Similar cu accesul la elementele unei structuri utilizand : nume_struct.nume_camp

EXEMPLU

Ex. 1: acces la elementele unei structuri tip camp de biti :

```
struct campbit
    { unsigned a:1;
      unsigned b:4;
      unsigned c:6;
     } A;
A.a=1;
```

Dispunerea in memorie a celor 3 campuri de biti:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Campul c						Campul b				a

TEST kahoot

Pentru login, introduceti codul afisat pe ecran, in browser la adresa:

http://kahoot.it