INFORMATIKA LABORATÓRIUM I.

Összetett adatszerkezetek

A C nyelv származtatott adatszerkezetei

- struktúrák,
- struktúrák bitmezőkre,
- unionok,
- enumerációk
- felhasználó által definiált változók.

Struktúrák

A struktúra egy vagy több, esetleg különböző típusú változó együttese, amelyeket az egyszerű kezelhetőség érdekében gyűjtünk össze. Ezekre egy névvel hivatkozhatunk.

A struktúra deklaráció alapvető formája:

```
struct  struktúra címke {
    típus változónév;
    típus változónév;
            .
    };
```

Ennek eredményeképpen a fordító helyet nem foglal, csak egy újabb változótípust hoztunk létre. A továbbiakban ezt a típust úgy használhatjuk, mint bármelyik alaptípust.

Struktúra definíciója

struct ***struktúra címke struktúra vált1, . .. , struktúra váltn;***

Ez a sor **n** darab struktúra címke típusú struktúrát definiál.

A struktúrák deklarálásának és definiálásának ajánlott módja:

```
struct  struktúra címke{
            típus változónév;
            típus változónév;
                .
                .
            } struktúra változók;
```

Azaz a deklarációt lezáró rész után azonnal következik a definiciós rész.

A mezőket, amelyek a struktúrát alkotják a struktúra elemeinek nevezzük. Az egy struktúrába tartozó elemek logikailag összetartoznak.

Példa

1.

```
struct date{
    int day;
    int month;                    -----struktúra tag(ok)
    int year;
};
```

ez utóbbi zárójelet változólista követheti, amely azt jelenti, hogy ugyanolyan típusként definiálja, mint a date-t.

2.

```
struct{
    char name[30];
    int state[3];
    unsigned long int zip;
    }addr-info;
```

Ebben az esetben a deklaráció nem történt meg, csak a definíció. Ez a C-ben legális, de nem ajánlott, mert a későbbiekben ugyanilyen struktúrát már nem definiálhatunk.

Struktúra változó deklarálása a következőképpen is történhet:

Hivatkozás a struktúra elemeire

A struktúra elemeire oly módon hivatkozhatunk, hogy a struktúraváltozó nevét követi a . (pont) operátor, majd ezt követi a struktúra-tag neve.

Formája:

struktúra változó.elem

Például:

```
addr_info.zip=12345;
printf("%d",addr_info.zip);
gets(addr_info.name);
```

Az addr_info.name a name struktúra elem kezdetére mutató karakter-pointer. Az addr_info.name egyedi karaktereire a következőképpen hivatkozhatunk:

```
register t;
for(t=0;addr_info.name[t];++t)
    putchar(addr_info.name[t]);
```

Struktúrák inicializálása

Struktúrák inicializálása a tömbök inicializálásához hasonlóan történhet. A kívánt kezdeti értékeket kapcsos zárójelek között soroljuk fel. Az értékek típusban feleljenek meg a struktúra tag típusának.

Példa:

```
struct date datum = { 18, 6, 1987 };
```

Struktúra tömbök

A struktúrákat leggyakrabban struktúra tömbökben használjuk fel. Egy struktúra tömb deklarálása úgy történhet, hogy deklarálunk először egy struktúrát, majd egy ilyen struktúra típusú tömböt.

Például:

```
struct addr addr_info[100];
```

Ez létrehoz egy olyan 100 elemű tömböt, amelynek elemei úgy vannak szervezve, mint az addr típusú struktúra. A tömb harmadik elemének ZIP kódjára például a következő módon hivatkozhatunk:

```
printf("%d",addr_info[2].ZIP);
```

Struktúra pointerek

A C nyelv lehetővé teszi struktúrára mutató pointerek használatát a változók pointereihez hasonlóan.

Például:

```
struct  addr   *addr_pointer;
```

Az addr_pointer változót egy addr típusú struktúrára mutató pointerként definiáljuk.

A struktúra pointerek felhasználási területe:

- a hivatkozás szerinti függvényhívást megvalósítani.
- a C dinamikus allokáló rendszerével szerkesztett listákat és más dinamikus adatstruktúrákat megvalósítani.

Amikor egy struktúra pointert adunk át egy függvénynek, akkor csak a struktúra címe kerül átadásra, így sokkal gyorsabb a függvényhívás, mint amikor egy struktúrát adunk át.

Példa:

```
struct bal  {
        float balance;
        char name [ 80 ];
        } person;
struct bal *p;              / *struktúra pointer deklarálása */
p = &person;                / * a person struktúra címét elhelyezi a p pointerbe */
```

Az előző példák definícióit és deklarálásait alapul véve, az alábbi kifejezések jelentéseit tekintsük át:

(*p).balance	a balance elemre hivatkozás a zárójelek szükségesek, mert a . (pont) operátor magasabb prioritású mint a *
p-> struktúratag;	ahol p a struktúrát megcímző mutató
struct date d;	úgy definiálja a d változót, hogy az d típusú struktúra legyen
struct date *pd;	pd olyan mutató, amely date típusú struktúrára mutat
++ (p -> x)	
++ p -> x	x-et inkrementálja, nem p-t
(++p) -> x	p-t inkrementálja

Tömbök és struktúrák elhelyezése struktúrákban

- Egy struktúra elemei lehetnek egyszerűek és összetettek.
- Egyszerű elem egy tetszőleges adattípus, pl. egész vagy karakteres lehet.
- Egy összetett adattípus pedig pl. egy tömb, vagy egy másik struktúra lehet.
- Az utóbbi esetben beszélünk egymásba ágyazott struktúrákról.

Példa:

1.

```
struct x {
        int a [10] [8];
        float b;
        } y;
```

Az y struktúra a tömbjének egy eleme: y.a[3][7].

2.

```
struct emp {
        struct addr address;
        float wage;
        } worker;
```

Unionok

A union egy olyan memóriaterület, amelyet több különböző változó használ. Ezek a változók különböző tipusúak lehetnek. A union deklarálása a struktúráéhoz hasonlóan történik, csak union kulcsszó használatával..

Példa:

```
union u {
        int i;
        char ch;
        };
```

Ez a struktúra deklarációjának megfelelően nem definiál változókat. Változó definiálása a deklaráció végén vagy egy külön utasítással valósítható meg a következőképpen:

```
union u cnvt;
```

- A cnvt union-ban az i int és ch char típusú változó ugyanazt a helyet foglalja el.
- A union deklarálásakor a compiler automatikusan a union-ban szereplő legnagyobb típusú változó számára szükséges helyet foglalja le.
- A union elemeinek elérésére ugyanaz a szintaxis érvényes, mint a struktúra elemeinek elérésére, azaz a . és -> operátort kell használnunk.
- Ha a union elemeire közvetlenül hivatkozunk, akkor használjuk a . operátort, ha az elemeket pointeren keresztül érjük el, akkor pedig a -> operátort.

Példák:

1. cnvt.i=10; a cnvt i elemének értéke 10 lesz.
2.
```
func1(un)
union union_tipus *un;
{
        un - = 10;
}
```

3.
```
union number  {
        int integer;
        float decimal;
}number;
union  number  data;
data.integer= 20000;
printf ("int: %6d,  float: %f10.4\n", data.integer,data.decimal);

data.decimal = 123.0;
printf ("int: %6d,  float: %f10.4\n", data.integer,data.decimal);
}
```

A program a következőket írja ki:

```
20000          0.0000
31553        123.0000
```

Enumeráció

Az enumeráció megnevezett egész konstansoknak egy együttese, amely specifikálja az összes olyan értéket, amelyet ez a változó felvehet.

Az enumeráció deklarációja:

enum enumeráció típusa {enumeráció lista} változó lista;

Példa: az USA-ban használatos érmék enumerációja

```
enum coin {penny, nickel , dime, quarter, half-dollar, dollar};
enum coin money;
```

Az enumerációban mindegyik szimbólum egy egész értéket képvisel, így ezeket egész kifejezésekben használhatjuk fel. Inicializálás nélkül az első szimbólum értéke 0, a másodiké 1, és így tovább.

Azaz a

```
printf ("%d %d", penny, dime);
```

kiírja a "0 2" értékeket a képernyőre.

Az enumerációban szereplő szimbólumokat inicializálhatjuk. Inicializálva, az inicializálás utáni szimbólumok értékei az inicializálás helyétől az inicializált értéktől kezdődően növekednek.

Például:

```
enum coin {penny, nickel, quarter=100, half-dollar, dollar};
```

A szimbólumok értékei:

penny	0
nickel	1
quarter	100
half-dollar	101
dollar	102

Típusnév definíciók

A typedef segítségével új típust hozhatunk létre.

Alapvető formája:

typedef típus név

ahol a típus, bármely C-ben engedélyezett típus lehet, a név pedig az új típus név.

Például:

```
typedef  int  LENGTH;
```

A LENGTH az int szinonímája lesz. A LENGTH a típus deklarációkban ugyanúgy használható, mint az int kulcsszó.

Azaz

```
LENGTH      len,  maxlen;
LENGTH      *lenght();
```

Struktúra példa

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <string.h>
# define letszam 3

int main()
{
  struct tanulo{
    char nev[30];
    int jegy;
  };
  struct tanulo web[letszam];
  int i;
  //feltoltes
  for (i = 0; i <= letszam-1; i++) {
    printf("Tanulo neve: ");
    scanf("%s", &web[i].nev);
    printf("Tanulo jegye: ");
    scanf("%d", &web[i].jegy);
    }

getch();
}
//---------------------------------------------------------------------------
```