C Programlama Dili'ne Giriş

Ders 15: *Yapılar ve Birlikler*

`###################- (%95)`

En son güncelleme: Wed, 30 Nov 2011 13:22:02 +0200

Giriş

C, kullanıcının kendi veri tipini tanımlamasına müsaade eder. Bu kısımda böyle veritiplerinin nasıl oluşturulacağı anlatılacaktır.

15.1 `enum` Deyimi (Enumeration Constants)

Bu tip, değişkenin alabileceği değerlerin belli (sabit) olduğu durumlarda programı daha okunabilir hale getirmek için kullanılır. Genel yazım biçimi:

```
enum tip_adı{değer_1, değer_2, ..., değer_n} değişken_adı;
```

tip_adı programcı tarafından verilen tip ismidir. *değişken_adı* ise program içinde kullanılacak olan değişkenin adıdır. Eğer kullanılmazsa program içinde daha sonra `enum` ile birlikte kullanılır. Örneğin:

```
enum bolumler{programcilik, donanim, muhasebe, motor};
```

tanımı ile derleyici `programcilik` için 0, `donanim` için 1, `muhasebe` için 2 ve `motor` için 3 değerini kabul ederek atamaları buna göre yapar. Değişken adı bildirilirse daha sonra `enum` kullanmaya gerek kalmaz. Örneğin:

```
enum renkler {kirmizi, mavi, sari} renk;
enum gunler  {pazartesi, sali, carsamba, persembe, cuma, cumartesi, pazar};
```

gibi yapılan sabit tanımlamaları program içinde kullanılabilir:

```
enum bolumler bolum;
enum gunler gun;

...

bolum = muhasebe; /* bolum = 2 anlamında */
gun   = cuma;     /* gun   = 4 anlamında */
renk  = kirmizi;  /* renk  = 0 anlamında */
```

Program 15.1 `enum` anahtar kelimesinin basit kullanımları gösterilmiştir.

Program 15.1: *enum kullanımı*

```
01: /* 15prg01.c: Klavyeden girilen bir sayının tek olup olmadığını sınar */
02:
03: #include <stdio.h>
04:
05: enum BOOLEAN{ FALSE, TRUE };  /* 0, 1 */
06:
07: int tek(int n){ return (n % 2);  }
08:
09: int main()
10: {
11:     enum BOOLEAN sonuc;
12:     int x;
13:
14:     printf("Bir sayi girin: ");
15:     scanf("%d",&x);
16:
17:     sonuc = tek(x);   /* tek mi? */
18:
19:     if( sonuc == TRUE )
20:          puts("Girilen sayi tek ");
21:     else
22:          puts("Girilen sayi cift");
23:
24:   return 0;
25: }
```

ÇIKTI

```
Bir sayi girin: 5
Girilen sayi tek
```

`enum` bloğu içinde tanımlanmış değişkenlerin sahip sabit olacağı değerler Program 15.2 olduğu gibi programcı tarafından belirlenebilir.

Program 15.2: *enum kullanımı*

```
01: /* 15prg02.c: Beş sabit bölüm için enum kullanımı */
02:
03: enum bolumler{
04:    programcilik = 1,  /* 1 */
05:    donanim,           /* 2 */
06:    muhasebe,          /* 3 */
07:    motor,             /* 4 */
08:    buro               /* 5 */
09: } bolum;
10:
11: int main()
12: {
13:     bolum = donanim;
14:
15:     printf("bolum : %d\n",bolum);
16:
17:     bolum += 2;  /* bolum = motor */
18:
19:     printf("Yeni bolum : %d\n",bolum);
20:
21:  return 0;
22: }
```

ÇIKTI

```
bolum : 2
Yeni bolum : 4
```

15.2 Yapı, `struct` Deyimi (Structures)

Aralarında mantıksal bir ilişki bulunan farklı türden bilgiler "yapılar (structures)" içerisinde mantıksal bir bütün olarak ifade edilebilir. Yapılar diziler gibi bellekte sürekli kalır. Bir yapı içerisindeki elemanlara üye (member) denir. Üyelerin herbiri farklı veri tipine sahip olabilir. Bu sayede, kendi tipinizi üretebilirsiniz.

Genel yapı bildirimi:

```
struct yapı_adı{
   tip yapı_değişken_ismi;
   tip yapı_değişken_ismi;
      ...
};
```

> **NOT**
>
> `enum` ile sabit bildirimi yapılırken
> `struct` ile değişken bildirimi yapılır.

Bir öğrenciye ait bilgileri bir çatı altında aşağıdaki gibi toplanabilir:

```
/* kayit veri tipi! */
struct kayit{
   /* üyeler */
   char  ad[10];
   long  no;
   short sinif;
};
```

Bu tipte bir değişken tanımlama:

```
struct kayit ogr1, ogr2;
```

şeklinde olabilir. `ogr1` değişkeni ile tanımlanan 1. öğrencinin numarasına bir değer atama işlemi:

```
ogr1.no = 2012597;
```

şeklinde yapılır. Bu deyimin kullanımı Program 15.3, ve 15.4'de gösterilmiştir. Inceleyiniz.

Program 15.3: *struct deyiminin kullanımı*

```
01: /* 15prg03.c
02:    Bir öğrenciye ait bilgilerin struct deyimi ile bir çatı altında toplanması  */
03:
04: #include <stdio.h>
05:
06: /* kayit yapısı */
07: struct kayit{
08:     char ad[10];
09:     long no;
10:     int  sinif;
11: };
12:
13: int main()
14: {
15:    struct kayit ogr;   /* ogr değişkeni kayit tipinde */
16:
17:    printf("Ogrenci Nosu  : "); scanf("%ld",&ogr.no);
18:    printf("Ogrenci Adi   : "); scanf("%s" , ogr.ad);
19:    printf("Ogrenci Sinifi: "); scanf("%d" ,&ogr.sinif);
20:
21:    printf("\n*** Girilen bilgiler ***");
22:    printf("\nNo    : %ld",ogr.no);
```

```
23:    printf("\nAdi    : %s ",ogr.ad);
24:    printf("\nSinifi: %d ",ogr.sinif);
25:
26:  return 0;
27: }
```

ÇIKTI

```
Ogrenci Nosu  : 948589
Ogrenci Adi   : Ahmet
Ogrenci Sinifi: 2

*** Girilen bilgiler ***
No    : 948589
Adi   : Ahmet
Sinifi: 2
```

Yapılar, diğer değişkenler gibi, fonksiyonlara parametre olarak geçirelilebilir.

Program 15.4: *struct deyiminin kullanımı*

```
01: /* 15prg04.c
02:    Yapıların bir fonksiyona parametere olarak aktarılabilmesi */
03:
04: #include <stdio.h>
05:
06: struct TARIH{
07:     int gun,ay,yil;
08: };
09:
10: void goster(struct TARIH x){
11:     printf("Tarih: %02d-%02d-%4d\n", x.gun, x.ay, x.yil);
12: }
13:
14: int main()
15: {
16:    struct TARIH n;  /* n değişkeni TARIH tipinde */
17:
18:    n.gun = 1;
19:    n.ay  = 8;
20:    n.yil = 2003;
21:
22:    goster(n);
23:
24:  return 0;
25: }
```

ÇIKTI

```
Tarih: 01-08-2003
```

Yapılarla da, gösterici tanımlamaları yapılabilir. Ancak, bir yapı gösteren göstericinin, gösterdiği yere yönlendirme operatörü, ->, ile erişilir.

Program 15.5: *struct deyiminin kullanımı*

```
01: /* 15prg05.c: Bir yapı gösteren gösterici */
02:
03: #include <stdio.h>
04: #include <stdlib.h>
05:
06: struct Meyve{
07:     float agirlik;
08:     float fiyat;
09: };
10:
11: int main()
12: {
13:    struct Meyve *muz, elma;
14:    float muzTutar, elmaTutar;
15:
16:    /* muz Meyve tipinde bir gösterici */
17:    muz = (struct Meyve*) malloc( sizeof(struct Meyve) );
18:    muz->agirlik = 2.50;
19:    muz->fiyat   = 3.50;
20:    muzTutar = muz->fiyat * muz->agirlik;
21:
22:    /* elma Meyve tipinde bir değisken */
23:    elma.agirlik = 2.00;
24:    elma.fiyat   = 1.75;
25:    elmaTutar = elma.fiyat * elma.agirlik;
26:
27:    printf("Meyve    Agırlık  Birim Fiyatı     TUTAR (TL)\n");
28:    printf("-----    -------  ------------    -----------\n");
29:
30:    printf("Muz      %7.2f       %7.2f        %7.2f\n",
31:            muz->agirlik, muz->fiyat, muzTutar);
32:
33:    printf("Elma     %7.2f       %7.2f        %7.2f\n",
34:            elma.agirlik, elma.fiyat, elmaTutar);
35:
36:  return 0;
37: }
```

ÇIKTI

```
Meyve    Agırlık  Birim Fiyatı     TUTAR (TL)
-----    -------  ------------    -----------
```

```
Muz         2.50          3.50          8.75
Elma        2.00          1.75          3.50
```

15.3 `typedef` Deyimi

`struct` ile oluşturulan yapıda `typedef` deyimi kullanılırsa, bu yapıdan değişken tanımlamak için tekrar `struct` deyiminin kullanılmasına gerek kalmaz.

```
typedef struct kayit{
  char  ad[10];
  long  no;
  short sinif;
} ogr1,ogr2;
```

Program 15.4 küçük bir değişiklike Program 15.6'de yeniden yazılmıştır. Inceleyiniz.

Program 15.6: *typedef - struct deyiminin kullanımı*

```
01: /* 15prg06.c
02:    Yapıların bir fonksiyona parametere olarak aktarılabilmesi.
03:    typedef deyimi kullanıldığında struct deyimine gerek yoktur */
04:
05: #include <stdio.h>
06:
07: typedef struct{
08:     int gun,ay,yil;
09: }TARIH;
10:
11: void goster(TARIH x){
12:     printf("Tarih: %02d-%02d-%4d\n", x.gun, x.ay, x.yil);
13: }
14:
15: int main(void)
16: {
17:    TARIH n;
18:
19:    n.gun = 1;
20:    n.ay  = 8;
21:    n.yil = 2003;
22:
23:    goster(n);
24:
25:  return 0;
26: }
```

ÇIKTI

```
Tarih: 01-08-2003
```

`typedef` başka kullanımı da vardır. C dilinde program kodları bu deyimle tamamen türkçeleştirilebilir. Örneğin bu deyim:

```
typedef int tamsayi;
```

şeklinde kullanılırsa programda daha sonra `int` tipinde bir değişken tanımlarken şu biçimde kullanılmasına izin verilir.

```
tamsayi x,y;   /* int x,y anlaminda */
```

15.1 Birlik, `union` Deyimi

Birlikler de yapılar gibi sürekli belleğe yerleşen nesnelerdir. Birlikler yapılara göre seyrek kullanılır. Bir programda veya fonksiyonda değişkenlerin aynı bellek alanını paylaşması için ortaklık bildirimi union deyimi ile yapılır. Bu da belleğin daha verimli kullanılmasına imkan verir. Bu tipte bildirim yapılırken `struct` yerine `union` yazılır. Genel yazım biçimi:

```
union birlik_adı{
   tip birlik_değişken_ismi;
   tip birlik_değişken_ismi;
      ...
};

union paylas{
      float f;
      int   i;
      char  kr;
};
```

Yukarıdaki bildirim yapıldığında, değişkenler için bellekte bir yer ayrılmaz. Değişken bildirimi:

```
union paylas bir,iki;
```

şeklinde yapılır. Üyelere erişmek aşağıdaki gibi olur:

```
bir.kr= 'A';
iki.f = 3.14;
bir.i = 2000;
```

Program 15.7: *typedef - deyiminin kullanımı*

```
01: /* 15prg07.c
02:    union x ve y nin aynı bellek alanını işgal ettiğinin kanıtı */
```

```
03:
04: #include <stdio.h>
05:
06: union paylas{
07:    int x;
08:    int y;
09: }z;
10:
11: int main()
12: {
13:    int *xAdres,*yAdres;
14:
15:    z.x    = 11;
16:    xAdres = &z.x;
17:    printf("x = %d   y = %d\n",z.x, z.y);
18:
19:    z.y    = 22;
20:    yAdres = &z.y;
21:    printf("y = %d   y = %d\n",z.x, z.y);
22:
23:    printf("xAdres = %p   yAdres = %p\n",xAdres, yAdres);
24:
25:  return 0;
26: }
```

ÇIKTI

```
x = 11   y = 11
y = 22   y = 22
xAdres = 0x804974c   yAdres = 0x804974c
```

Powered by PHP