enum, struct, union ve typedef Yapıları

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

enum

Bu tip, değişkenin alabileceği değerlerin belli(sabit) olduğu durumlarda programı daha okunabilir hale getirmek için kullanılır. enum örnekleri Program 1 ve Program 2 de gösterilmiştir. Genel yazım biçimi:

```
enum TipAdı{değer1,değer2,...,değerN}DeğişkenAdı;
```

`TipAdı` programcı tarafından verilen tip ismidir. `DeğişkenAdı` 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.

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

Şimdi istenirse tanımlanan bu tipler program içinde kullanılabilir.

```
enum bolumler bolum;
enum gunler gun;

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

Program 1 : *3 sabit renk için enum kullanımı*

```
1:  /* enum1.c */
2:  enum renkler{kirmizi,sari,mavi};
3:
4:  main(){
5:       enum renkler renk;
6:       renk = sari;
7:       printf("\n%d",renk);
8:  }
```

Program 2 : *5 sabit bölüm için enum kullanımı*

```
1:  /* enum2.c */
2:  enum bolumler{
3:    programcilik, donanim,   muhasebe, motor, buro
4:  }bolum;
5:
6:  main(){
7:     bolum = donanim;
8:     printf("\n%d",bolum);
9:     bolum +=2;  /* bolum=motor */
10:    printf("\n%d",bolum);
11: }
```

struct

C dilinde standart tipler kullanılarak kendi tipinizi üretebilirsiniz. (struct örnekleri Program 3 , Program 4 ve Program 5 te gösterilmiştir). Bu yapının kullanımı:

```
struct TipAdı{
    tip deg_ismi;
    tip deg_ismi;
    ...
};
```

enum ile sabit bildirimi yapılırken struct ile değişken bildirimi yapılır. Bu yapının faydası, örneğin bir öğrenciye ait bilgileri bir çatı altında toplamak için:

```
struct ogrenci{
    char  ad[10],soyad[20];
    long  no;
     short sinif;
};
```

Bu tipte bir değişken tanımlama:

```
 struct ogrenci ogr1,ogr2;
```

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

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

veya

```
 ogr1->no = 2012597;
```

şeklinde yapılır.

Program 3 : *Bir öğrenciye ait bilgilerin tek bir çatı altında toplanması*

```
1:  /* struct1.c */
2:  #include <stdio.h>
3:
4:  struct ogrenci{
5:      char ad[10],soyad[20];
6:      long no;
7:      int  sinif;
8:  }
9:
10: main(){
11:   struct ogrenci ogr;
12:   printf("Ogrenci nosu :");
13:   scanf("%ld",&ogr.no);
14:   if( ogr.no == 2248 )
15:   {
16:      ogr.no    = 2248;
17:      strcpy(ogr.ad,"Ahmet");
18:      strcpy(ogr.soyad,"Bingul");
19:      ogr.sinif = 1;
20:   }
21:   printf("\nNo    : %ld",ogr.no);
22:   printf("\nAdı   : %s ",ogr.ad);
23:   printf("\nSoyadı: %s ",ogr.soyad);
24:   printf("\nSınıfı: %d ",ogr.sinif);
25: }
```

Program 4 : *Bir öğrenciye ait bilgilerin tek bir çatı altında toplanması*

```
/* struct2.c */
#include <stdio.h>
struct ogrenci{
    char ad[10],soyad[20];
    long no;
    int  sinif;
}ogr;

main(){
   printf("Ogrenci nosu :");
   scanf("%ld",&ogr->no);
   if( ogr.no == 2248 )
   {
      ogr.no    = 2248;
      strcpy(ogr.ad,"Ahmet");
      strcpy(ogr.soyad,"Bingul");
      ogr.sinif = 1;
   }
   printf("\nNo    : %ld",ogr.no);
   printf("\nAdı   : %s ",ogr.ad);
   printf("\nSoyadı: %s ",ogr.soyad);
   printf("\nSınıfı: %d ",ogr.sinif);
}
```

Program 5 : *bir topun x-y düzlemindeki zamana bağlı hareketi*

```
1:  /* struct3.c */
2:  #include <stdio.h>
3:  #include <math.h>
4:
5:  struct koordinat{
6:    float x,y;
7:  }top;
8:
9:  main(){
10:   int i;
11:   float t;
12:
13:   for(t=0.0;t<10.0;t+=0.1){
14:     top->x = 10 - 9*cos(t);
15:     top->y = 5  + 2*sin(t);
16:     printf("%f\t%f",top->x,top->y);
17:   }
18: }
```

tpyedef

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

```
typedef struct kayit{
      char  ad[10],soyad[20];
      long  no;
      short sinif;
}ogr1,ogr2;
```

bu kullanımın diğerlerinden farkı yoktur.

Bu deyimin başka kullanımı da vardır. C dilinde program kodları bu deyimle tamamen türkçeleştirilebilir. Örneğin:

```
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 */
```

union

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.

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

Dikkat: Yukarıdaki bildirim yapıldığında, 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 6 : *union x ve y nin aynı bellek alanını işgal ettiğinin kanıtı*

```
1:  /* union1 */
2:  #include <stdio.h>
3:
4:  union paylas{
5:       int x;
6:       int y;
7:  }z;
8:
9:  main()
10: {
11:   int *X,*Y;
12:
13:    z.x = 10;
14:    X = &z.x;
15:    printf("\nTamsayı(x) : %d - bellek adresi %X",z.x,X);
16:
17:    z.y = 20;
18:    Y = &z.y;
20:    printf("\nTamsayı(y) : %d - bellek adresi %X",z.y,Y);
21: }
Tamsayı(x) : 10 - bellek adresi DF23
Tamsayı(y) : 20 - bellek adresi DF23
```