STRUCT – ENUM – UNION – TYPEDEF (YAPILAR)

By Mustafa Onur Parlak

- **Modelleme:** Problem çözümleri, olaylara yakınlaşma durumları gibi gerçek hayat ile bağlantılı fiziksel durumları; tasarım ortamına aktarılması işlemini anlatmaktadır.
- **Bu noktada** C dilini kullanarak **yapısal modelleme** yaparız.
- **STRUCT**
    - **Struct** ve **diziler** birbirine örtüşen belli tanımlamalara sahiptir.
        - **Veri topluluğunu ifade etmektedir.**
            - **(Dizi, aynı tip – Yapı, farklı tip olabilir)**
        - **İçerdiği elemanlar belleğe ardışıl yerleşir**
            - **(Dizi, indeks – Yapı, isim ile)**
    - **Struct,** birçok **kavramı** tek bir **çatı** altında toplayan **modelleme** aracıdır.
    - **Buraya kadar ki süreçte, FARKLI VERİ TİPLERİ olayı dikkat edilmesi gereken bir husustur.**
    - **Yapısal programlamada** karşımıza çıkan en büyük problemlerden birisi de, **struct için tanımlanan adres genişliğinin, ilgili platform ile uyuşmamasıdır.**
    - **Bu durumda "struct alignment problem"** meydana gelir ve yapıdaki elemanlar lüzumsuz yere **shift ve soyutlanmalara** girer.

```c
/* Bu struct tanımlamasında,
*  Etiket (Tag) olarak : 'Etiket' kullanılmış
*  Struct elemanı olarak : 'Sayi' ve 'Fiyat' kullanılmış
*
*  Elemanların farklı veri tipinde olduğuna dikkat edin.
*/
struct Etiket{
    int16_t Sayi;
    float_t Fiyat;
}Etiket;

struct Etiket kravat, gomlek;

int main ()
{
    struct Etiket Kiyafet;
    Kiyafet.Fiyat = 30;

    return 0;
}
```

- **TYPEDEF – STRUCT**
  - Kişisel **veri tipi** tanımlayıcısıdır. **Tekil** veya **çoklu veri tipi** tanımlarında kullanılabilir.
  - İhtiyaç duyulan; **genel amaç ile halledilemeyecek durumlar, özel durumlara özel bilgi setleri** gibi durumları **modellerken** ihtiyacımızı karşılamış oluruz.

```
/* Okunabilirliği arttırmıştır
*  Total değişiklikleri hızlı optimize eder
*/
typedef int Integer;
Integer nOgrenci, OgrenciNumarasi;
```

```
// Fazladan 'struct' yazmadık
typedef struct Etiket{
    int16_t Sayi;
    float_t Fiyat;
}Etiket;

Etiket kravat, gomlek;
```

- **Struct – Struct Ataması**
  - **İki Struct** modülünü birbirine atarken en sık yapılan hata, **pointer** durumdaki struct yapısını atamaktır. Eğer **iki struct'ın** aynı adresi göstermesini istiyorsanız (ki gerek yok) sorun olmayacaktır.

```
/* Soru    : İki adet sensör. Biri asil diğeri yedek.
*            Aynı işlevde çalışıyor.
*            Asil bozulursa, yedek devreye girecek
*
*  Gelişme : Sensör kodları her biri için yazılacak
*            Herhangi bir hatada diğerinin aynı kodla
*            aktif edilmesi
*
*  Sorun   : Sensörlerin kodları modüler olduğu için,
*            Tek tek kopyala mümkün değil
*
*  Sonuç   : Asil ve yedek için Yapısal sınıflandırma yapılacak.
*            Asil ve yedek için yapısal kopyası oluşturulacak.
*/
typedef struct Sensors{
uint8_t uPriority;
char cSensorName [ NAME_LEN + 1 ] ;
double_t SensorValues;
}Sensors;

Sensors Asil, Yedek;

// 1. Yöntem
Asil.uPriority = Yedek.uPriority;
strncopy(Asil.cSensorName, Yedek.cSensorName, (NAME_LEN + 1));
Yedek.SensorValues= Yedek.SensorValues;

// 2. Yöntem
memcpy(&Asil,&Yedek, sizeof(Sensors));

// 3. Yöntem
Asil = Yedek;
```

  - **1. Yöntem :** Daha fazla işlem hacmi ve kıyaslamaya tabii olan bu sistem örneği, ilgili iş için fazla uzundur.
  - **2. Yöntem :** İki belleğin işaret ettiği veya tuttuğu bellekteki uzunluğu kadar byte kopyalama işlemidir. **Hard Copy** olarak tanımlanan bu yol ile iki aynı değeri farklı adreslerde taşımış oluruz.
  - **3. Yöntem :** Basit gibi gözükse de, her şeyden önce operatör kullanılması yeterince işlemciyi yormaktadır. Bunun yanında derleme sırasında veya işlem sırasında optimize edilebileceği için sorunlara yol açmaktadır. Kopyalama yerine atama yapılmaktadır.
    Sonuç olarak, Asil devreden çıktığında, yedek de çıkacaktır.

- **Struct Adreslemesi**

```
printf("Asil Sensorun adresi :%d[%x]",&Asil,&Asil);
```

- **Struct – Pointer**
  - **Pointer,** herhangi bir şeyin **adresini** ve o adresteki **veriyi** tutabilir.
  - **Farklı türden** verileri ve veri türlerini işaret edebiliriz, fakat bu durum fazla byte tüketimine sebep olacağı için bu tarz çoklu durumlarda **struct** yapısını kullanırız. Yani **adres yerine bellek** kopyalarız.
    Bu sayede byte olarak daha az yer kaplamaktadır.
  - Önceki kısımlarda yer alan (**x.y)** kombinasyonu ile **struct** yapısındaki bellek alanına ulaşmayı sağlamaktadır.
    Fakat **pointer** olarak incelediğimizde, ***x.y** yazmak ile **x->y** yazmak arasında bir fark olmayacaktır.

```
/* Buradaki asıl olay,
*  Struct yapısı olarak tanımlanan pointer'ın
*  belleğe erişmek için, normal struct (.) ile
*  elemana ulaşamaz. Bu sebeple (->) kullanır
*  veya pointer olarak belirtir (*)
*/

// 1. Yöntem
(*sPointer).Value = 12;

// 2. Yöntem
sPointer->Value = 12;

// 3. Yöntem
double *pValue = &Asil.Value;
*pValue = 12;
```

- **Struct – Bit Erişimi**
  - Struct ile byte erişimi çok küçüktür. Ekstra olarak içteki elemanların byte'larını da küçülttüğümüzde **bellek** ve **modellemeyi** daha elverişli yapabiliriz.
  - Alt eleman tanımlamalarında byte aralığını değiştirebildiğimiz gibi aynı veri tipinde yazmamız **struct alignment** problemine yakalanmamızı önlemektedir.

- **ENUM**
  - **Veri tipidir. Tanımlanan değerler RAM'de tutulur.**
  - **Etiketlenmiş sayı yığını olarak adlandırılır.**
  - **Önceden belirlenmiş sabit değerler ile kullanılır.**
  - **İsim-Değer ikilisi** ile tanımlanır. Yani değerle değil, isimle iş yapılır.

```
/* Mevcut elemanlara, farklı değer atabilir
*  Örneğin : gomlek = 2 gibi...
*  Özel bir değer atanmaz ise,
*  ileri = 0, geri = 1, sag = 2, sol = 3'tür.
*/

typedef enum uRobotYonu {
    ileri,
    geri,
    sag,
    sol
}uRobotYonu;

uRobotYonu Tespit = sag;
```

  - **Enum – Define**
    - **Enum,** veri tipidir. **Define,** derlemede alınan bilgidir
    - **Etiket için 'define' kullanınız. Sayı yığını belirtmek için 'enum' kullanınız.**

- **UNION**
  - **Struct** ile fazlasıyla benzer yanları vardır.
    - **Struct:** Mevcut elemanlarını ardışıl sıralar ve belleği kaplar.
      Union: Mevcut eleman boyutu en fazla kendisi kadardır.
    - **Union:** Mevcut elemanları **aynı adresi** gösterir. Böylece verinin bilgiye dönüştürülmesi sağlanmış olur