Nedir Şu POD Meselesi?..

30-Mayıs-2009

POD (Plain Old Data) C++'ta ayrıntıları pek bilinmeyen karışık bir konudur. Pek çok dökümanda *POD* terimi geçtiği halde pek az *C++* programcısının bu konuyu tam olarak bildiğini görüyorum.

C'de genel olarak (ister *int*, *long* gibi aritmetik türden olsun, isterse dizi ya da yapı türünden olsun) her türden nesnenin byte'ları bellekte ardışıl bir biçimde tutulmaktadır. Nesnenin adresini alarak o adresten itibaren *sizeof* değeri kadar byte'ı *memcpy* gibi bir fonksiyonla bir alana aktarıp aynı biçimde geri alırsak eski nesneyi elde ederiz. *C* standartları bunu garanti etmektedir. Örneğin *T* herhangi bir türü belirtmek üzere:

```
#define N sizeof(T)
char buf[N];
T obj;
...
memcpy(buf, &obj, N);
memcpy(&obj, buf, N);
```

Fakat *C++* bunu garanti etmiyor. Yani *C++*'ta her türden nesnenin bellekte ardışıl byte topluluğu olarak bulunması garanti değil. *C++* standartları yalnızca *POD* diye tanımladığı türlere ilişkin nesnelerin byte'larının bellekte ardışıl olacağını garanti ediyor. Yani biz *C*'de herhangi bir nesneyi bir dosyaya tek hamlede *fwrite* gibi bir fonksiyonla aktarıp, *fread* gibi bir fonksiyonla geri alabilirken *C++*'ta bunu yalnızca *POD* özelliği taşıyan türler için yapabiliriz. *POD* sözcüğü *"Plain Old Data"* sözcüklerinden kısaltmadır. *POD C++*'ta tıpkı *C*'deki gibi byte'ları bellekte ardışıl olarak bulunmak zorunda olan nesneleri belirtmek için kullanılıyor.

Peki *C++*'ta *POD* tanımı nasıl yapılmış? Standartlar *POD* türlerini şöyle tanımlıyor: "Skaler türler (yani aritmetik türler ve adres türleri), *POD-struct* ve *POD-union* türleri, bu türlerden diziler ve bunların *const-volatile* biçimleri *POD* türlerdir." Yani standartlara göre *int*, *long*, *int* *, long *, *int* türden, *long* türden diziler *POD* nesnelerdir. Fakat gördüğünüz gibi standartlar "her sınıf (yapılar da sınıftır) *POD* türündendir" demiyor. Yalnızca *POD-struct* ve *POD-union* türlerinin *POD* olduğunu söylüyor. O halde bizim *POD-struct* ve *POD-union* türlerinin ne olduğunu anlamamız gerekiyor.

Standartlarda bir türün *POD-struct* olması için şu koşullar öngörülmüş:

- *POD-struct* bir toplaşık (*aggregate*) sınıftır.- *POD-struct static* olmayan (*non-static*)

- *POD-struct*, *POD-union* türünden olmayan veri elemanına ve referans veri elemanına sahip olamaz.

- *POD-struct* programcı tarafından tanımlanmış kopya atama operatör fonksiyonuna ve bitiş fonksiyonuna (*destructor*) sahip olamaz (toplaşık sınıflar zaten başlangıç fonksiyonuna sahip olamaz).

POD-union için de benzer koşullar öngörülmüş

- *POD-union* bir toplaşık birliktir.

- *POD-union static* olmayan (*non-static*) *POD-struct*, *POD-union* türünden olmayan veri elemanına ve referans veri elemanına sahip olamaz.

- *POD-union* programcı tarafından tanımlanmış kopya atama operatör fonksiyonuna ve bitiş fonksiyonuna (*destructor*) sahip olamaz (toplaşık sınıflar zaten başlangıç fonksiyonuna sahip olamaz).

Toplaşık tür (*aggregate type*) olma koşulları şöyledir:

- Sınıf ya da dizi türleri toplaşık türlerdir.

Eğer sınıf türü söz konusuysa:

- Sınıfta kullanıcının bildirdiği bir başlangıç fonksiyonu olmamalıdır.

- Sınıfta *private* ya da protected bölümde bildirilmiş *static* olmayan veri elemanı bulunmamalıdır.

- Sınıfın taban sınıfı olmamalıdır.

- Sınıfın sanal fonksiyonu bulunmamalıdır

Bu kadar çok tanımlamalardan sonra aklınızın karışmaması olanaksız :-). O halde ben konuyu ayrıntıları gözardı ederek şöyle basitleştireyim: Bir sınıfın ya da birliğin *POD* olabilmesi için onun taban sınıfının olmaması, sanal fonksiyonunun olmaması, programcı tarafından tanımlanan bir başlangıç fonksiyonunun, kopya atama operatör fonksiyonunun ve bitiş fonksiyonunun olmaması gerekir. Ayrıca *POD* bir sınıf ya da birlik yalnızca *POD* türden *static* olmayan *public* veri elemanlarına sahip olabilir. Zaten böyle bir sınıf *C*'deki yapı gibi bir sınıftır. Böyle sınıflar türünden nesnelerin byte'ları ardışıl olmak zorundadır.

POD kavramının ilkdeğer verilme işlemi üzerinde de etkisi vardır. Standartlara göre sınıf nesnesi yaratılırken eğer parantezler kullanılmamışsa ve eğer sınıf *POD* değilse nesne için sınıfın *default* başlangıç fonksiyonu çağrılır. Fakat sınıf *POD* ise, herhangi bir ilkdeğerleme işlemi yapılmaz.

```
X a;
X *p = new X;
```

Burada eğer *X*, *POD* ise sınıfın *default* başlangıç fonksiyonu çağrılır fakat *POD* değilse hiçbir başlangıç fonksiyonu çağrılmaz.

Gördüğünüz gibi standartlara göre *C++*'ta bir sınıfın elemanları ardışıl bir biçimde bulunmak zorunda değildir. Gerçekten de standartlarda sınıfların anlatıldığı bölümde sınıfın iki erişim belirleyici anahtar sözcüğü arasındaki kısmın ardışıl olması gerektiği belirtilmiş fakat bütünsel olarak bu garanti verilmemiştir. Fakat standartlar pek çok

olası ortam göz önüne alınarak hepsinin içerebileceği ortak özellikler bağlamında oluşturulmuştur. Pratikte baktığımızda ise yaygın olarak kullanılan derleyicilerin hepsinin sınıfların veri elemanlarını ardışıl bir biçimde yerleştirdiğini görmekteyiz. (Bu nedenle standartlardaki durumu fazlasıyla ciddiye alıp bu konuyu bilgiçlik taslama amacıyla kullanmayınız :-)). En iyisi ben size şöyle yardımcı olayım: Siz elinizdeki sınıfa ilişkin nesnenin byte'larını ardışıl varsayın. Ben kefilim. Eğer bu nedenden dolayı sorun çıkarsa bana bildirin, ben çözmeyi garanti ediyorum :-).

Ayrıca *C++0x*'te *POD* kavramının biraz daha genişletildiğini belirteyim. Örneğin taban sınıf *POD* ise türemiş sınıfın *POD* olabilmesi gerekirdi. Fakat *C++2003* bunu yasaklamış. (*C++0x*'teki *POD* kavramı hakkında geçen sene makale formunda birşeyler yazmıştım. Onu da yakında makaleler kısmına yerleştiririm.)