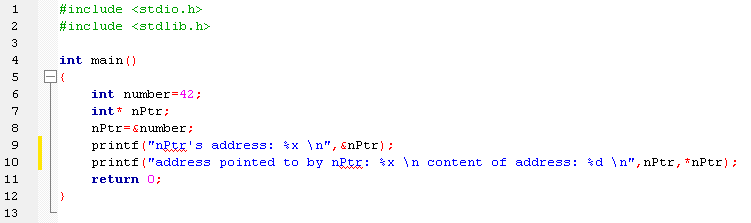
Pointer 101

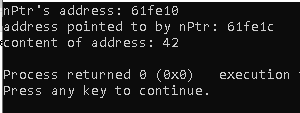
Pointer; C/C++ dillerinde önemli bir yere sahip ve uzmanlaşmamız için biraz efor sarfetmemiz gereken bir konudur. Ama temel mantığını anladığımız sürece bize oldukça kolaylıklar sağlayacaktır. Bu yazımızda, birlikte pointer ile ilgili önemli işlevleri ve o işlevleri kavramamıza yardımcı olacak örnekleri gözden geçireceğiz.

Pointer'ın Temel İşlevi  
  
Bilgisayar ortamındaki her bir veri değeri, hafızada belli bir yer kaplar: float ise 4 byte, char ise 1 byte, double ise 8 byte... Bir de değerlerin kapladığı alanı temsil eden adresler vardır. Adresin temsil ettiği bit değeri, mimariye göre değişiklik gösterir. Günümüzdeki bilgisayarların çoğunda ise her bir adres, 1 byte'lık (8 bit'lik) bir alanı temsil eder. Adresler onaltılık (hexadecimal) sisteme göre numaralandırılır.

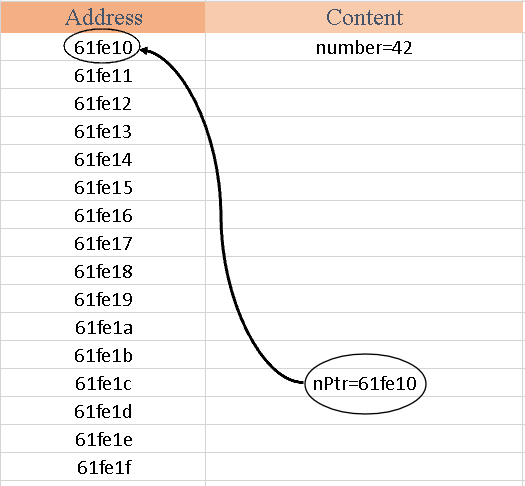
Nasıl ki temelde gördüğümüz her bir değişken bir değeri ifade eder, pointer da o değişkenlerin adreslerini ifade eder. Bunun daha iyi anlaşılması için aşağıdaki basit kodu ve tabloyu inceleyelim.



Burada 42 sayısını atadığımız number değişkeninin adresine işaret edecek bir pointer tanımlandığını görüyoruz. Tanımlama için öncelikle pointer’ın veri tipi ile işaret edeceği adresteki değişkenin tipi aynı olmalı. Bu, ileriki aşamalarda bahsedeceğimiz pointer aritmetiği ile ilgili işlemlerde de bellek hatalarının olmaması için önemlidir. Yıldız işareti (\*), işlemciye bir pointer atanacağını ifade eder. &number, number değişkenin adres değeridir. Ampersand (&), isminin başına geldiği değişkenin adresini belirtir. nPtr isimli pointer’a, number’ın adresini atadığımıza göre printf ile ekrana çıktı gönderebiliriz:

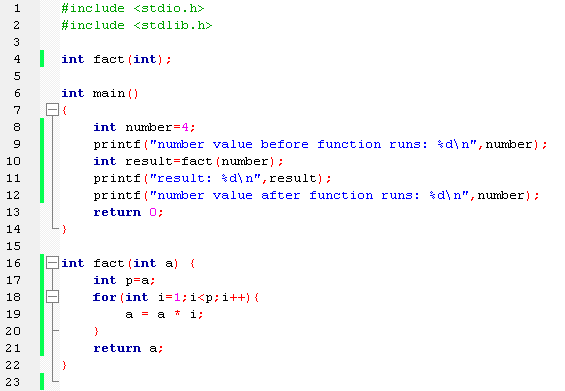


Gördüğümüz üzere nPtr pointer’ının bulunduğu adres, işaret ettiği adres ve o adresin içeriğindeki değeri ekrana bastırdık. Evet, pointer da aslında adresi olan bir değişkendir: Diğer bildiğimiz değişkenlerle arasındaki fark ise değerinin bir adresi ifade etmesidir. Bunu daha iyi kavramak için yukarıdaki kodda yaptığımızı basit bir tabloya aktarmak mümkün:

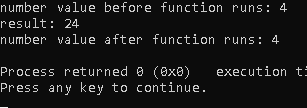


Tablo, kodda 61fe1c adresindeki pointer’a, number’ın bulunduğu 61fe10 adresinin atandığını gösteriyor. Yani pointer, adres depolayan bir değişken görevi görüyor.

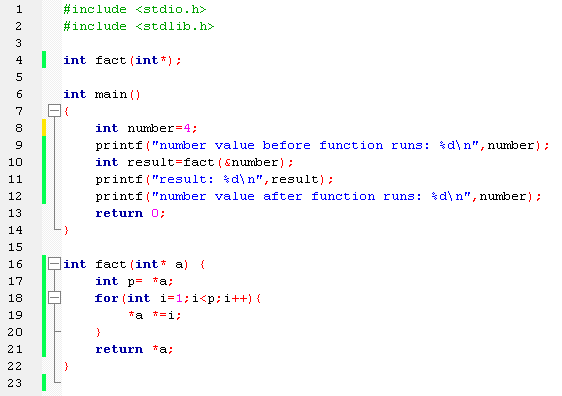
# Pointer ile Call by Reference (Referans ile Çağrı)

Call by value (değer ile çağrı) işleminde fonksiyonlar, main’den aldığı değişkenlerin değerini değiştiremez. Değişkenleri klonlayıp kopyaları üzerinde işlemler yapar. Ama orijinal değişkenler üzerinde hiçbir etkisi olmaz. Örnek olarak bir sayının faktöriyelini hesaplayan koda bakalım.

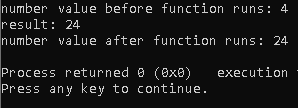
number değişkeninin bir kopyası fonksiyonda parametre olarak kullanıldı. Görüldüğü üzere fonksiyon içindeki komutlar, verilen parametrenin değerini değiştiriyor. Ama number değişkeni üzerinde hiçbir etkisi olmuyor. Kodumuzu çalıştırıp sonucun nasıl olacağını birlikte görelim:



Evet, number değişkeni üzerinde hiçbir değişiklik yok. Peki ya değişkenin adresini kopyalayıp parametre olarak kullansaydık? İşte bu, doğrudan adresin içindeki bağımsız değişkene erişmemize olanak tanır. Aynı kod üzerinde oynamalar yapalım:



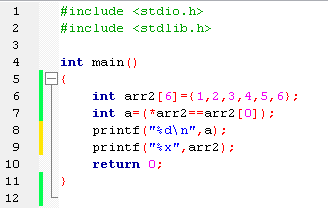
Gördüğünüz gibi buradaki fark, number’ın adres değerinin kopyasının fonksiyondaki pointer’a atanmasıdır.



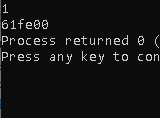
Kodumuzu çalıştırdığımızda number değerinin değiştiğini fark ediyoruz. Call by value ile call by reference arasındaki bu farkın nedeni, birden fazla değişkenin “aynı” içeriğe sahip olabilmesi ama her birinin “farklı ve tek” bir adreste bulunmasında gizli.

Pointer, Dizi ve Pointer Aritmetiği

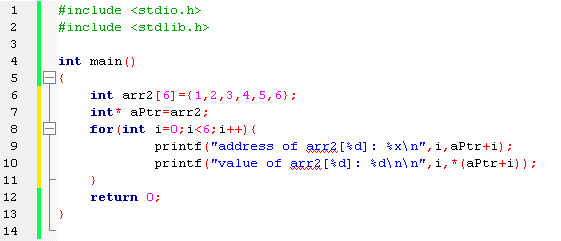
Diziler, birden fazla değeri temsil eden değişkenlerdir. Belirtilen değer sayısı kadar statik bellek tahsisi yapılır. Örneğin double tipinde dört değer tutan dizinin bellekte kapladığı alan 4xsizeof(double) = 32 byte olacaktır. Pointer ve dizinin belli başlı ortak yönleri vardır. Beraber inceleyelim.



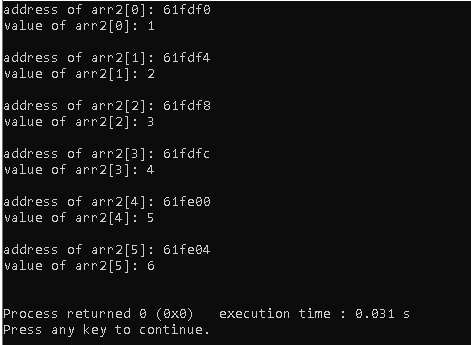
Diziyi tanımladıktan sonra dizinin ismi olan arr2, bir adresi ifade eder. Yukarıdaki a değişkenine atadığımız basit mantıksal işleme bakacak olursak sonuç, 1 olacaktır. Çünkü arr2, birinci elemanın adresine işaret eden bir pointer görevi görür. Bu yüzden birinci elemanın adresinin içeriğini ifade eden \*arr2 ile arr2[0], aynı değeri ifade eder. Son olarak arr2’nin tuttuğu adres değeri hexadecimal biçimde ekrana yazdırılır. Ekran çıktısı:



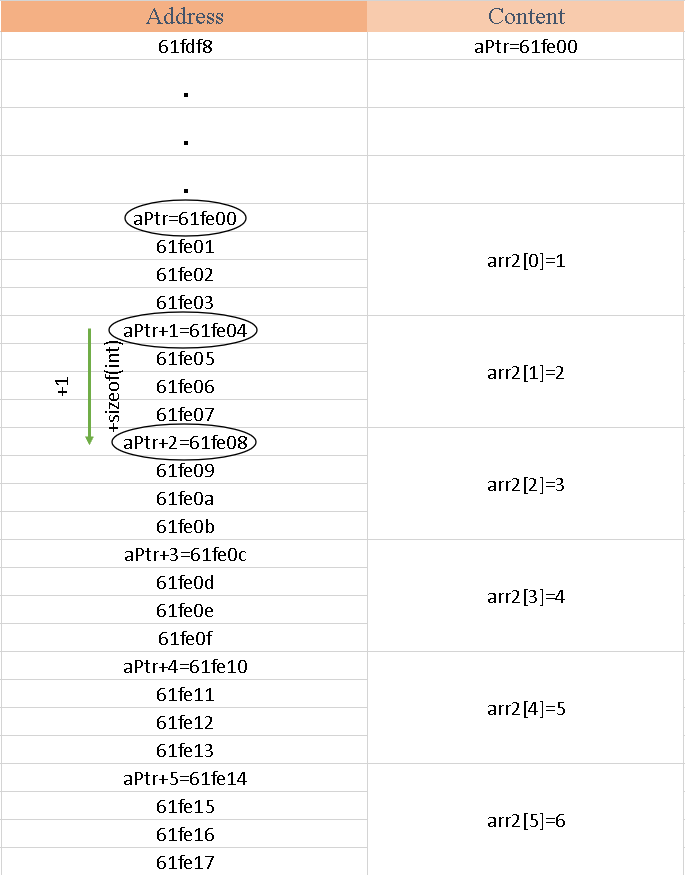
Bir de pointer aritmetiğine göz atalım. Hatırlarsanız, pointer’ın veri tipinin bu hususta önemli olduğunu belirtmiştim. Evet, başlarda da belirttiğimiz gibi günümüzdeki çoğu bilgisayarda adreslerin her biri 1 byte’lık bir alanı temsil eder. İnt tipinde tek bir değişken tanımlayacak olursak 4 byte’lık bir alan kaplanacağı için değişken dört adrese depolanır. Eğer int tipinde 6 elemanlık bir dizi tanımlayacak olursak dizinin kapladığı alan 24 byte olacağı için bu bellek bloğunun adres sayısı da 24 olacaktır. Her bir elemanı ise 4 adreslik alan kaplayacaktır ve başlangıç adresleri içerikteki değeri temsil edecektir. Pointer aritmetiği adresler arasında geçişler yapmamızı mümkün kılar. Mesela 6 elemanlık bu dizide birinci elemanın adres değeri a ise, ikinci elemanın adres değeri a+sizeof(int) olur. Bunun nedeni, birinci elemanın adresine işaret eden pointer, int cinsinde tanımlı. Bu da birim artış değerini int boyutuna çeviriyor. Aşağıdaki kod ve tabloyla daha iyi anlaşılacak:



Dizinin ilk elemanının adresini belirten arr2’yi, aPtr isimli pointer’a atadık (Bu atamayı yapmadan da dizinin ilk elemanın pointerı ile geçişler yapabileceğinizi unutmayın. Ben daha anlaşılır olması açısından bir pointer kullandım. Ama dizi ismi olan arr2, bir constant pointer olduğu için sonradan adres değerinin değiştirilemeyeceğini ve bu yüzden “arr2++” değil de sadece “arr2+(sayı)” şeklinde geçişler için kullanabileceğimizi de hatırlatmakta fayda var) Burada her bir dizi elemanının adres ve değerini sırasıyla ekrana yazdıracağız. Kodumuzu çalıştıralım:



Tablomuz yardımıyla bellekteki işlemleri gözden geçirelim:



Evet, aPtr int türünde olduğu için “aPtr + (sayı)” ifadesi, “aPtr + sizeof(int) \* (sayı)” ifadesine eş değer oluyor.