Bir fonksiyon kendini çağırabilir mi?



www.yzokul.com.tr

Bir fonksiyon kendini çağırabilir mi?



Bir fonksiyonun kendini çağırıp çağıramayacağını merak edebilirsin. Kısa yanıt evet. Bir fonksiyon kendini çağırdığında buna özyineleme denir. Bu kitap özyinelemeyi adım adım açıklar; fonksiyonun ne olduğundan başlayıp özyinelemeyi güvenli biçimde nasıl kullanacağına kadar gelir. Basit örnekler görecek ve neden bir durma noktası gerektiğini öğreneceksin.



Fonksiyon, çağırdığında bir işi yapan, adlandırılmış bir talimat bloğudur. Bir fonksiyonu çağırmak, ondan çalışmasını istemek demektir. Örneğin, karşılama mesajı yazdıran greet adında bir fonksiyonun olabilir. greet() yazdığında greet içindeki talimatlar çalışır. Fonksiyonlar kodu düzenlemene ve aynı talimatları yeniden yazmadan kodu tekrar kullanmana yardımcı olur.



Bir fonksiyonun sonuç almak için başka bir fonksiyonu çağırdığı basit bir örnek: Bir sayıyı ikiyle çarpan double adında bir fonksiyonun olduğunu varsayalım. Başka bir fonksiyon olan printDouble, double'ı çağırır ve sonucu yazdırır. printDouble(5) çağırdığında double(5) çağrılır, 10 elde edilir ve 10 yazdırılır. Bu, bir fonksiyonun başka kodu tetikleyebileceğini gösterir.



Şimdi fikri tersine çevir: Bir fonksiyonun gövdesi kendine bir çağrı içerebilir. Örneğin, bir sayıyı yazdırıp sonra countdown'ı daha küçük bir sayıyla çağıran bir countdown fonksiyonu. Bu kendini çağırma, özyinelemenin anlamıdır. Fonksiyonun özel bir anahtar sözcüğe ihtiyacı yoktur; gövdesinde yalnızca kendi adını kullanır.



Özyineleme, bir fonksiyonun kendini çağırmasının adıdır. Yeni ya da sihirli bir şey olmaz; bu, daha önce gördüğün aynı çağırma fikridir. Bir fonksiyon kendini çağırdığında bilgisayar bunu diğer fonksiyon çağrıları gibi ele alır: mevcut işi duraklatır, çağrılan fonksiyonu çalıştırır ve sonra devam eder. Tek fark, çağrılan fonksiyonun aynı fonksiyon olmasıdır.



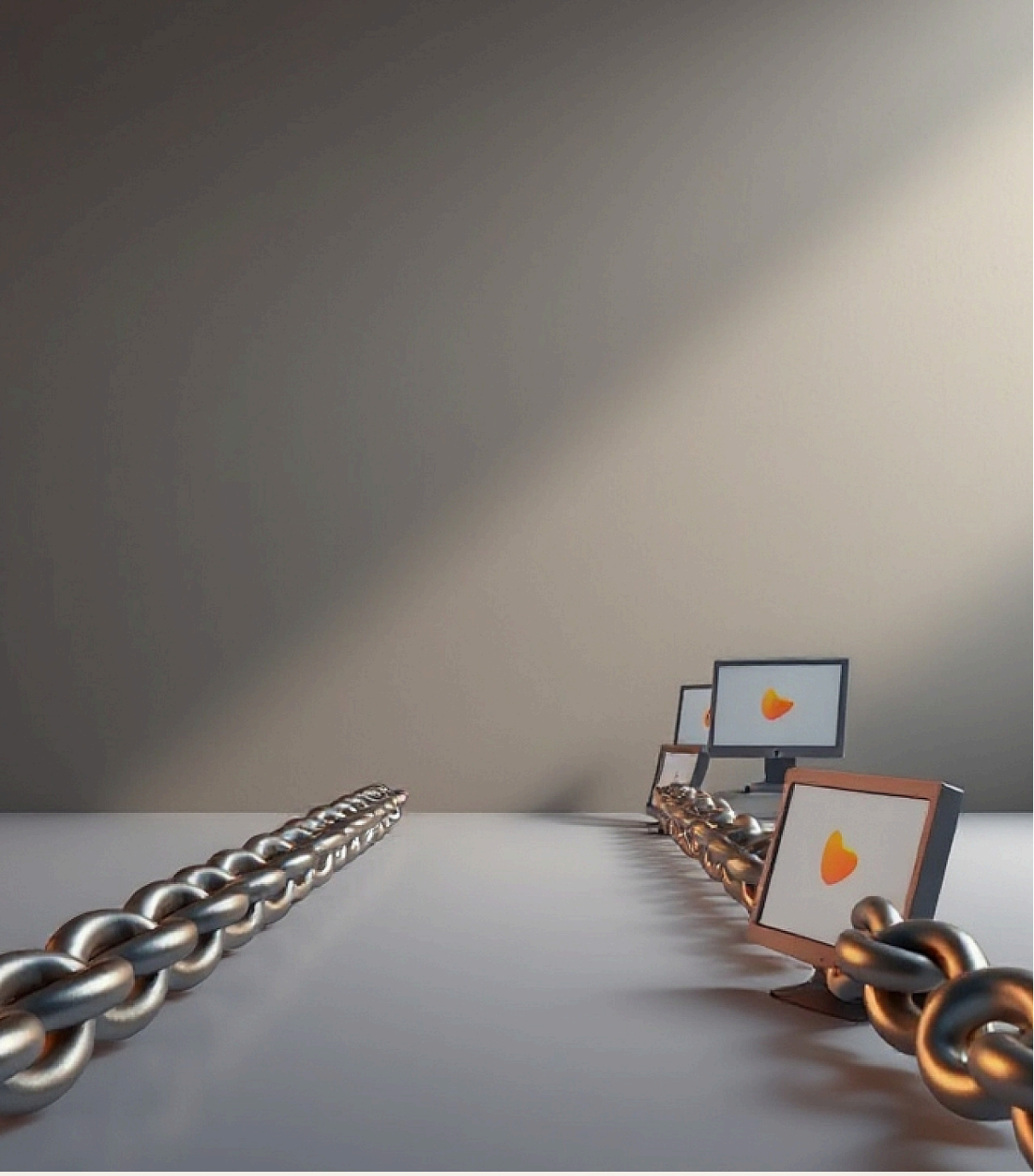
Özyineleme, bir problem aynı problemin daha küçük sürümlerini içerdiğinde işe yarar. Küçük bir kopyayı çözmek, büyük olanı çözmeye benzer. Örneğin, bir klasörde dosya bulmak her alt klasörü aramayı gerektirebilir; bu da aynı görevin daha küçük ölçekte yapılmasıdır. Fonksiyon kendini daha küçük parçalar üzerinde çağırarak iç içe yapıları doğal biçimde ele alabilir.



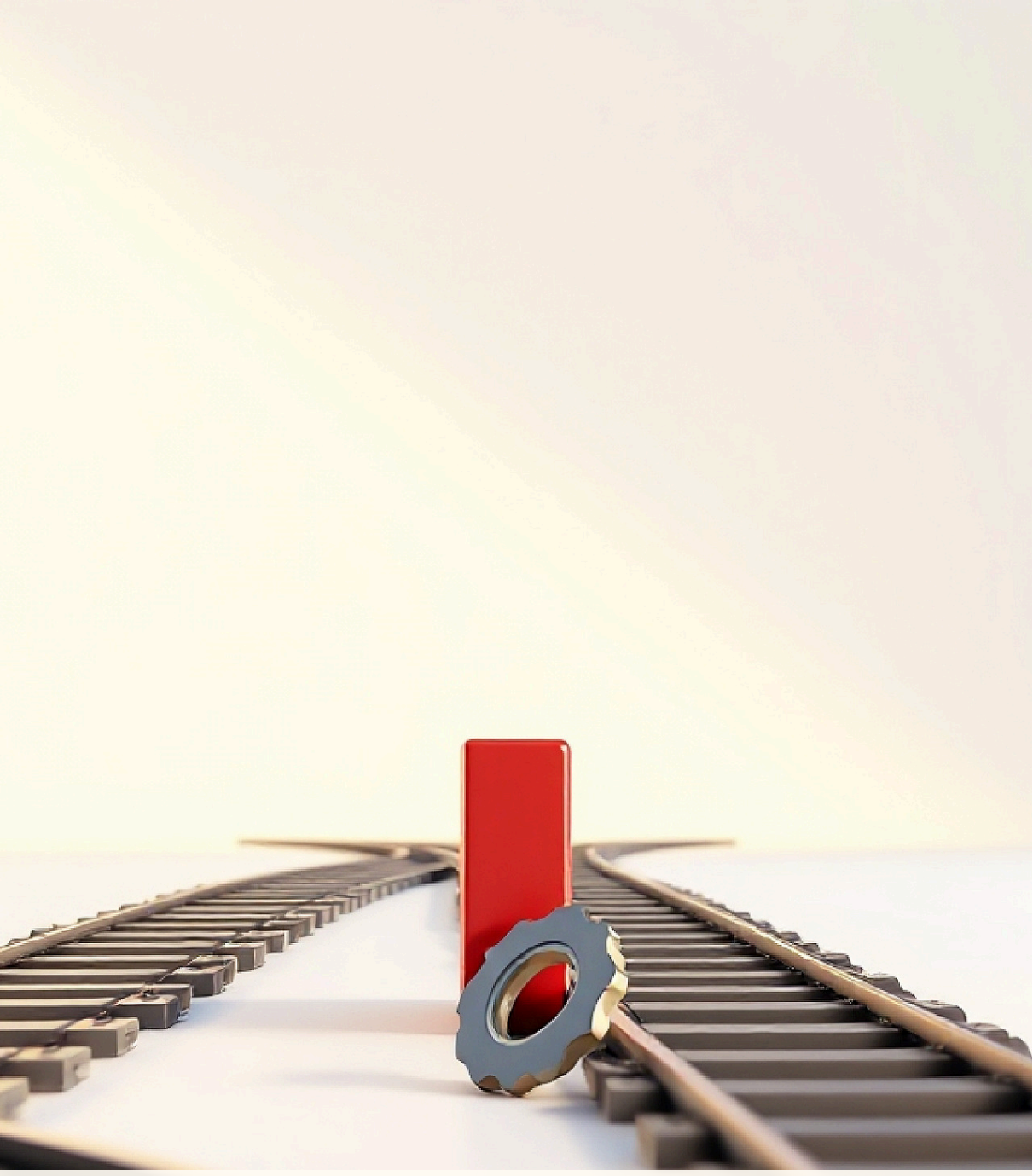
İçinde klasörler bulunan, onların da daha fazla klasör içerdiği bir klasör düşün. Tüm klasörün içeriğini kontrol etmek için her alt klasörü aynı şekilde kontrol etmen gerekir. Bu, klasörlerden oluşan bir ağaç gibidir. Aynı işlem—dosyaları listeleme ve alt klasörlerin içine bakma—her düzeyde tekrarlanır. Özyineleme bu kalıba çok uygundur.



Bir klasör kontrol fonksiyonu her alt klasörde kendini çağırabilir. Her çağrı daha küçük bir parça üzerinde aynı işi yapar: öğeleri listeler ve her alt klasör için fonksiyonu yeniden çağırır. Böylece fonksiyon, her düzey için ayrı kod yazmadan her derinlikteki iç içeliği ele alır. Aynı mantık her düzeyde geçerlidir.



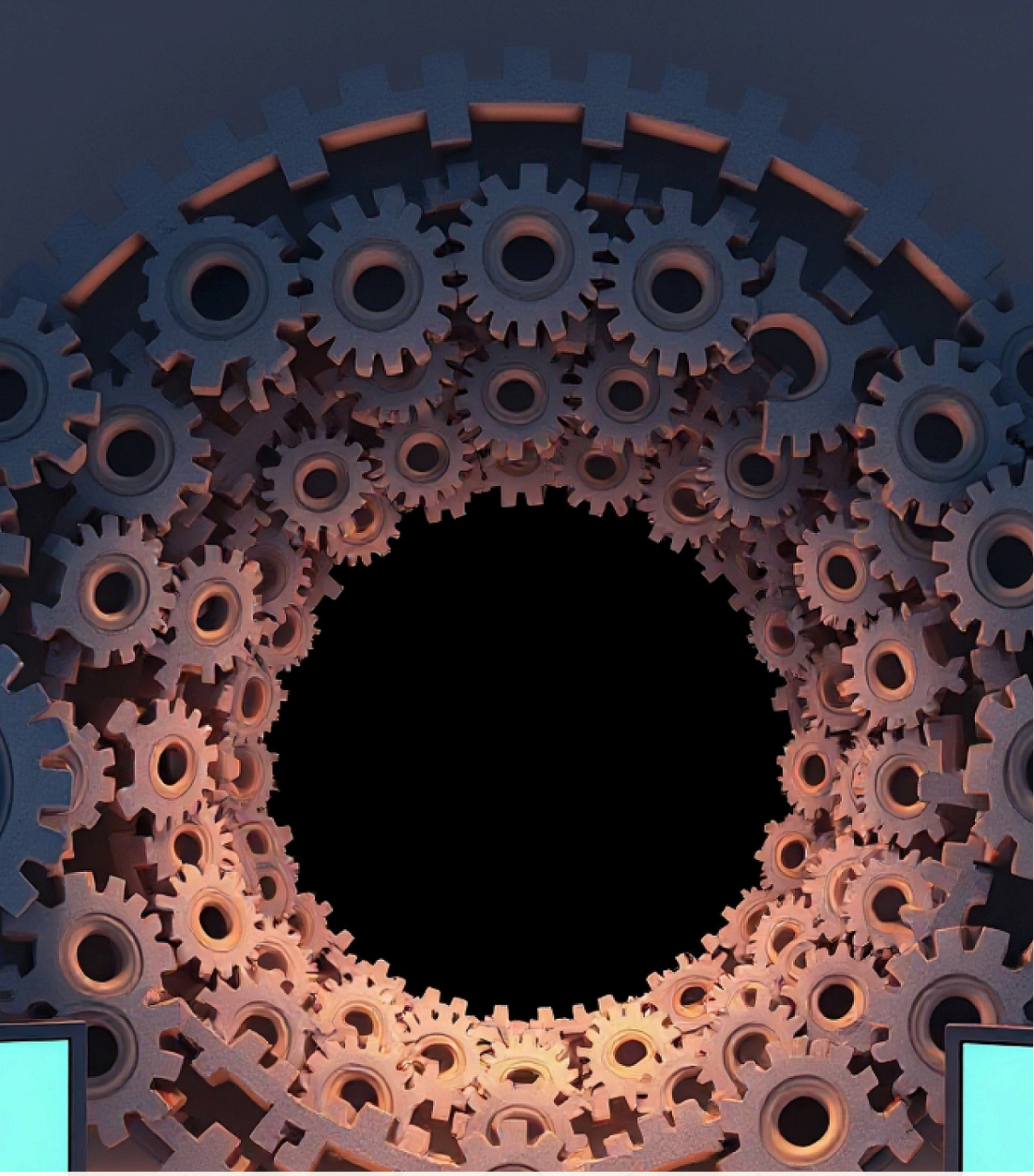
Kendini çağırmalar sonsuza kadar süremez. Fonksiyonun kendini çağırmayı bıraktığı bir nokta olmalıdır. Aksi halde fonksiyon sonu gelmeden kendini çağırmaya devam eder; hiç bitmeyen bir döngü gibi. Bunu önlemek için her özyinelemeli fonksiyonun en az bir durumda kendini çağırmaması gerekir. Bu duruma temel durum denir.



Temel durum, fonksiyonun kendini çağırmadan bir yanıt döndürdüğü basit durumdur. Çağrı zincirini bitiren şey budur. Örneğin, bir geri sayım fonksiyonunda temel durum, sayı sıfıra ulaştığında olabilir: o zaman fonksiyon durur ve döner. Temel durum olmadan fonksiyon kendini sonsuza kadar çağırmaya devam ederdi.



Açık bir temel durumu ve bir kendini çağırması olan küçük bir örneği izle. sumTo(n) fonksiyonu 1'den n'e kadar sayıların toplamını döndürür. n 1 ise 1 döndürür (temel durum). Aksi halde n artı sumTo(n-1) döndürür. sumTo(3) çağrısı sumTo(1)'e kadar iner, sonra yanıtlar yukarı doğru geri gelir: 1, sonra 3, sonra 6.



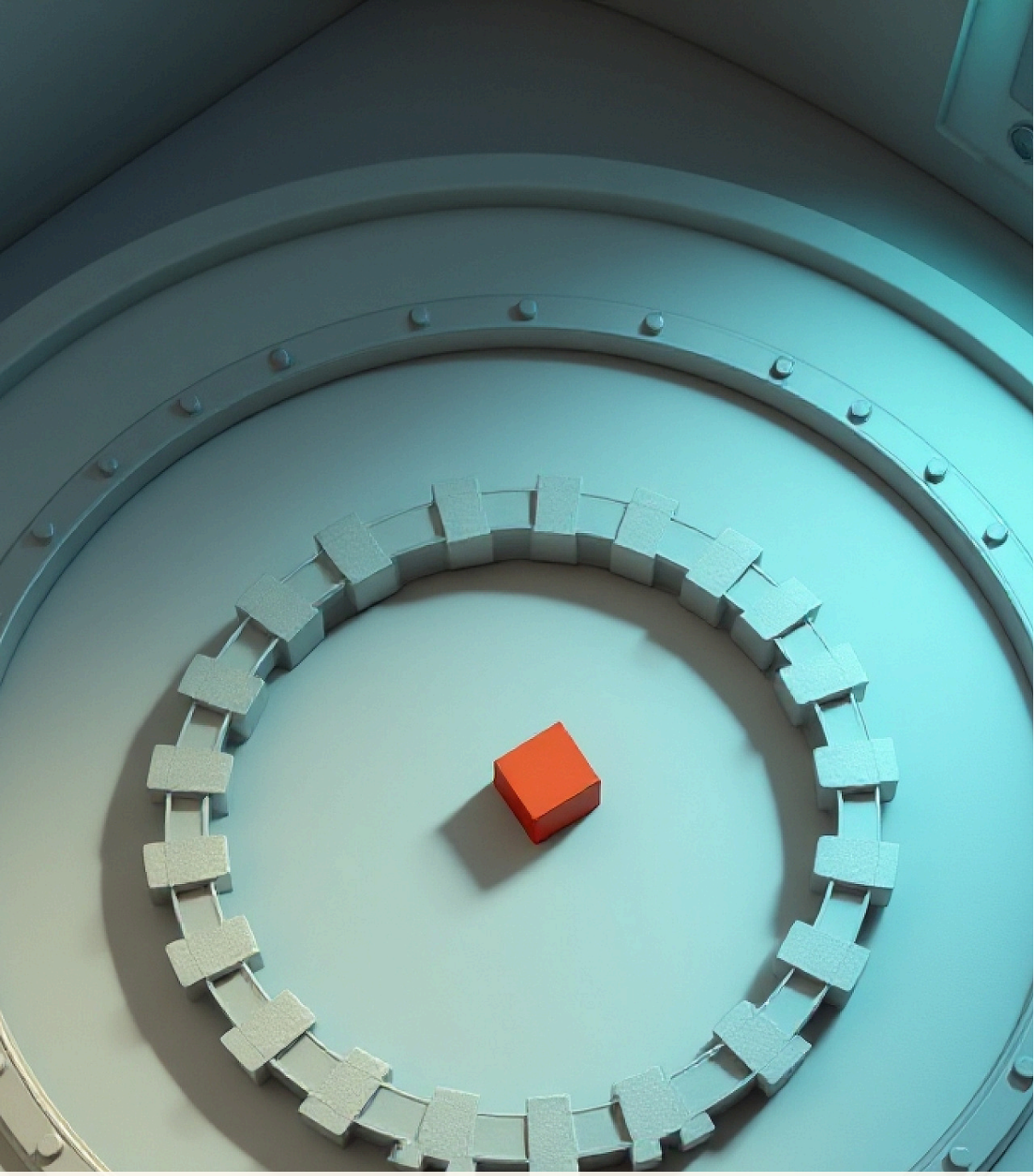
Temel durum eksik ya da yanlışsa çağrılar sonu olmadan devam eder. Program işini asla bitiremez çünkü her çağrı bir başka çağrı yapar. Örneğin, kendini her zaman aynı argümanla çağıran bir fonksiyon hiçbir zaman durma noktasına ulaşmaz. Bilgisayar, kaynakları tükenene kadar daha fazla çağrı çalıştırmayı sürdürür.



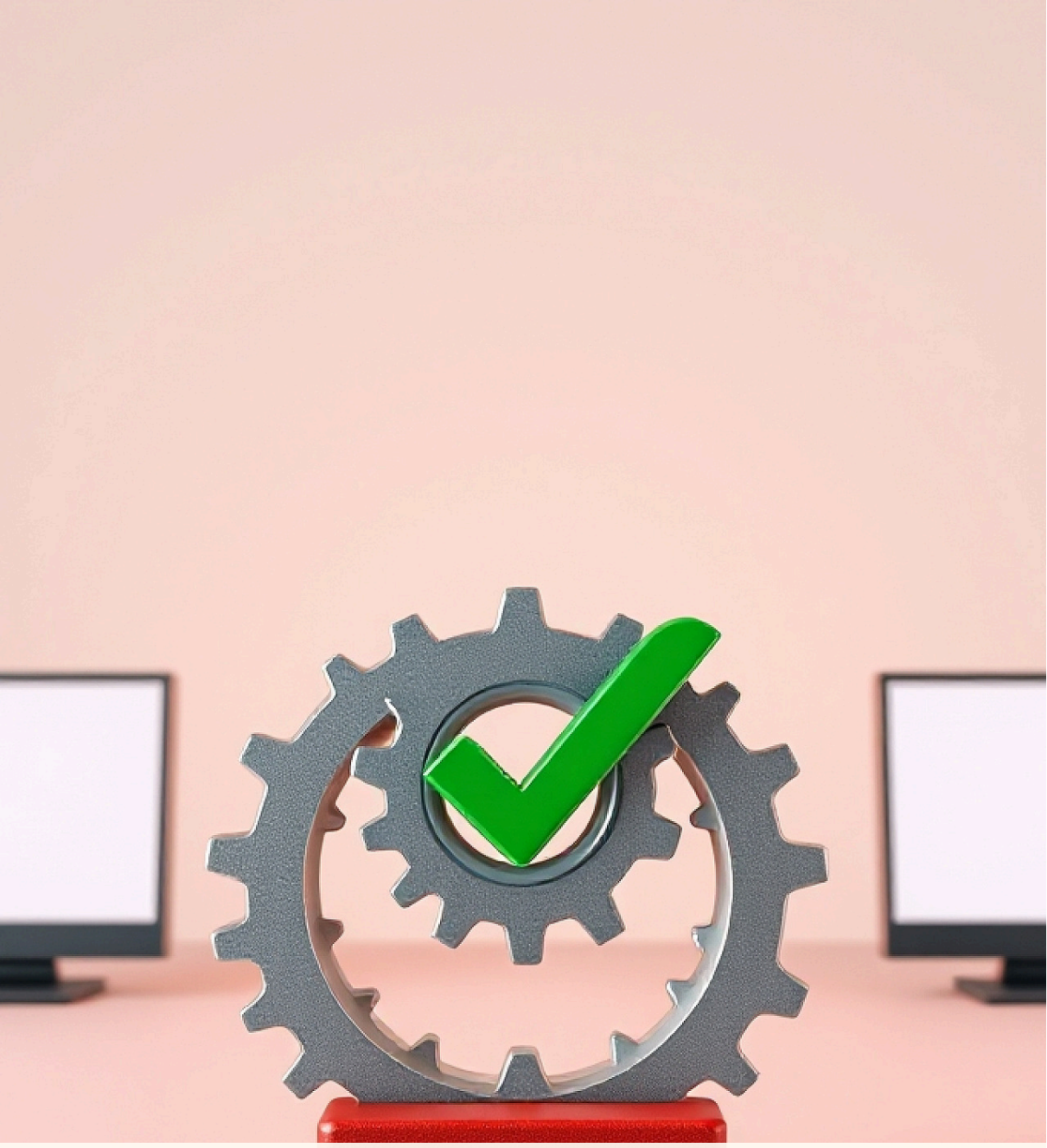
Sonsuz çağrılar sistemin sınırlı kaynaklarını tüketir. Her çağrı bellek ve işlem zamanı alır. Sonunda sistemin sınırlarına ulaşılır ve program başarısız olur; çoğu zaman yığın taşması gibi bir hatayla. Bu tehlikeli değildir ama programının çalışmasını durdurur. Bunu önlemek için her zaman bir temel durum ekle.



En yaygın hata, kendini çağırmayı yazıp temel durumu unutmaktır. Özyinelemeli bir fonksiyonu çalıştırmadan önce açık bir durma noktası olduğunu kontrol et. Kendine sor: En basit durum nedir ve fonksiyon orada kendini çağırmadan dönüyor mu? Dönmüyorsa o durumu ekle. Her özyinelemeli fonksiyonun en az bir temel duruma ihtiyacı vardır.



Temel duruma gerçekten daha küçük problemlerle ulaşılmalıdır. Problem ona hiç küçülmüyorsa çağrılar yine hiç durmaz. Örneğin, bir fonksiyon kendini daha küçük bir sayı yerine aynı sayıyla çağırırsa temel duruma asla ulaşılmaz. Her kendini çağırmanın problemin daha küçük ya da daha basit bir sürümü üzerinde çalıştığından emin ol.



Evet, bir fonksiyon kendini çağırabilir ve buna özyineleme denir. Bunu, iç içe klasörler gibi daha küçük benzer problemler içeren problemler için kullan. Çağrıların durması için her zaman bir temel durum ekle. Temel durum olmadan çağrılar, sistemin kaynak sınırlarına ulaşılana kadar devam eder. Özyinelemeli fonksiyonlarında açık bir durma noktası olup olmadığını kontrol et.