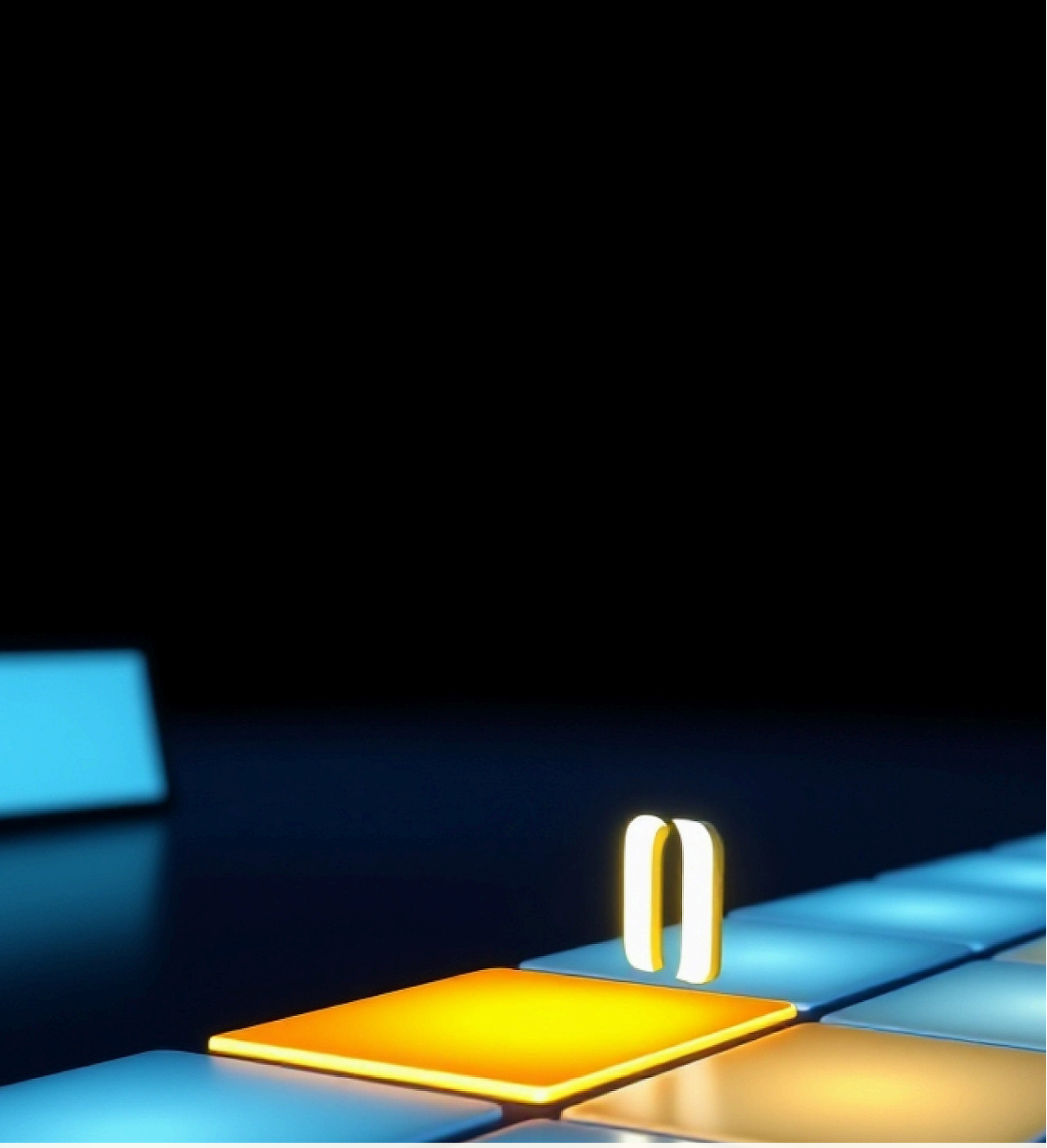
Kesme noktası ne işe yarar?



www.yzokul.com.tr

Kesme noktası ne işe yarar?



Kesme noktası ne işe yarar? Kesme noktası, çalışan bir programın seçilen bir satırda duraklamasını sağlar, böylece içine bakabilirsin. Bu kısa yanıt anahtardır: hata ayıklama sırasında programı belirli bir noktada durdurur. Böylece tahmin yürütmek yerine o anda neler olduğunu inceleyebilirsin. Hata ayıklama, bir programdaki hataları bulmak ve düzeltmek demektir.



Programlar çok hızlı çalışır. Bir program, art arda çok hızlı çalışan satırlardan oluşan bir listedir. Sen baktığında program çoktan bitmiştir, bu yüzden ortada neler olduğunu göremezsin. Bu hız hataları gizler. Bir hatayı bulmak için programı yavaşlatıp gözlemlemen gerekir. Kesme noktası tam da bunu yapan bir araçtır.



Hata ayıklama, bir programdaki hataları bulmak ve düzeltmek demektir. Bir hatayı düzeltmek için önce programın gerçekte ne yaptığını görmen gerekir. Göremediğin şeyi düzeltemezsin. Hata ayıklama araçları programın davranışını gözlemlemene yardımcı olur. Kesme noktası, programı duraklatıp yakından incelemene olanak tanıdığı için en kullanışlı hata ayıklama araçlarından biridir.



Bir programı video gibi düşün. Videoyu duraklatmak, tek bir kareyi ayrıntılı olarak incelemeni sağlar. Kesme noktası, bir program için duraklatma düğmesidir. Bir videoyu duraklattığında, hızla geçip gitmeden tek bir anı inceleyebilirsin. Benzer şekilde, kesme noktası programı durdurur, böylece yürütmesindeki tek bir anı inceleyebilirsin. Bu, neler olup bittiğini anlamayı kolaylaştırır.



Kesme noktası, kodun belirli bir satırına koyduğun bir işarettir. Program o satıra ulaştığında durur ve bekler. Programın duraklamasını istediğin satırı sen seçersin. Bu, bir kitaba yer imi koymaya benzer: o sayfaya geldiğinde durursun. Programlamada kesme noktası, hata ayıklayıcıya yürütmeyi o satırda durdurmasını söyler.



Program, işaretli satıra gelene kadar normal şekilde çalışır. Sonra devam etmek yerine tam o noktada donar. Kesme noktasından önceki her şey her zamanki gibi çalışır. Yürütme, kesme noktasının bulunduğu satıra ulaştığında program duraklar ve sırada ne yapacağını söylemeni bekler. Bu, program ilerlemeden etrafa bakma fırsatı verir.



Program duraklatılmışken değişkenlerinde saklanan değerlere bakabilirsin. Bu, o andaki programın iç durumunu gösterir. Değişkenler, veri tutan kaplar gibidir. Onları inceleyerek programın kesme noktasında neler bildiğini görebilirsin. Örneğin, score adlı bir değişkenin 10 olması gerekiyorsa, gerçekten 10 olup olmadığını kontrol edebilirsin.



Duraklatılmış bir program, bu noktadan önce hangi satırların çalıştığını görmeni de sağlar. Bu, programın bu noktaya nasıl geldiğini gösterir. Geçerli satıra götüren işlev çağrılarının listesi olan çağrı yığınına bakabilirsin. Bu, programın izlediği yolu anlamana yardımcı olur. Yolu bilmek çoğu zaman geçerli değerleri bilmek kadar önemlidir.



Küçük bir örnek düşün: bir program bir puan hesaplıyor ama sonuç yanlış. Hatayı yapan satırı bulmak istiyorsun. Programın çok sayıda satırı olabilir ve hata herhangi bir yerde olabilir. Kesme noktası kullanarak programı durdurup puanı farklı noktalarda kontrol edebilirsin. Bu, işlerin nerede ters gittiğini daraltır.



Kesme noktasını, yanlış puanın hesaplanmasından hemen önceki satıra koy. Program tam o hesaplama çalışmadan önce duraklar. Böylece hesaplamaya giren değerleri kontrol edebilirsin. Girdiler doğruysa hata hesaplamanın kendisinde olabilir. Girdiler yanlışsa hata daha önce olmuştur.



Duraklamadan itibaren kodu birer satır çalıştırırsın. Buna adımlama denir. Her satırdan sonra neyin değiştiğini görmek için değişkenleri yeniden kontrol edersin. Adımlama, programın mantığını yakından izlemeni sağlar. Hangi satırın bir değeri tam olarak nasıl değiştirdiğini görürsün. Bu, programın ne yaptığını tahmin etmekten çok daha kolaydır.



Adım adım, değerlerin değişmesini izler ve bir değerin yanlış hale geldiği anı yakalarsın. Bu seni hatalı satıra yönlendirir. Örneğin, total adlı bir değişkenin pozitif olması gerekirken aniden negatif olduğunu görebilirsin. Bu değişiklik belirli bir satırda gerçekleşir ve büyük olasılıkla hatanın kaynağı orasıdır. Bulduğunda düzeltebilirsin.



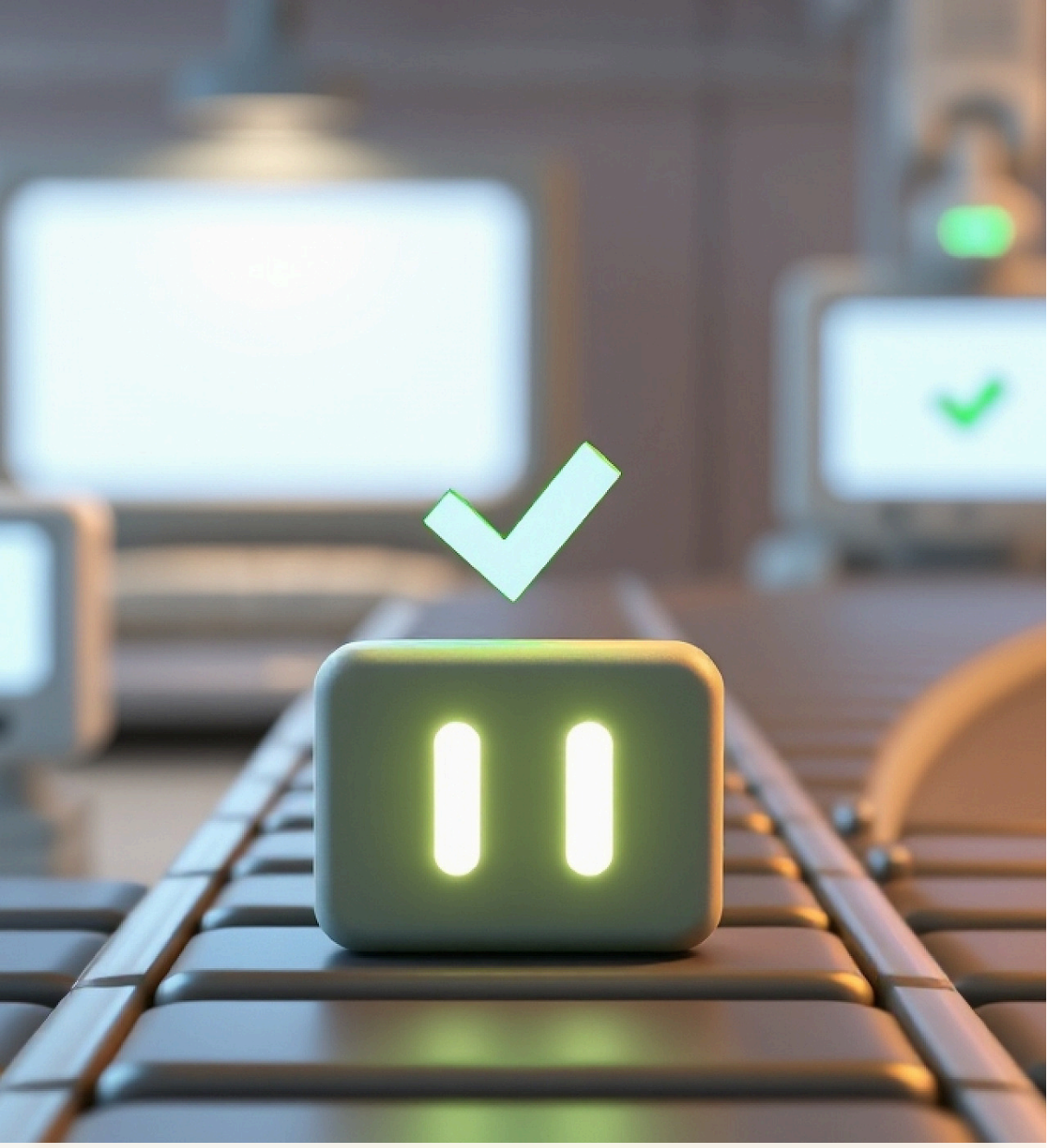
Yararı açıktır: hızlı bir program yavaş ve gözlemlenebilir hale gelir. Tahmin yürütmek yerine iç durumunu inceleyebilirsin. Kesme noktası olmadan değerleri göstermek için print deyimleri ekleyebilirsin, ancak bu kodu değiştirir ve dağınık olabilir. Kesme noktası, kodu değiştirmeden bakmanı sağlar. Zamana hâkim olmanı sağlar, böylece her anı inceleyebilirsin.



Yaygın bir hata, aynı anda birçok satıra kesme noktası koymaktır. Sonra sürekli durur ve izini kaybedersin, bu yüzden iyi seçilmiş tek bir satırla başla. Çok fazla kesme noktası hata ayıklamayı kafa karıştırıcı hale getirir çünkü program sık sık durur. Sorunun olduğundan şüphelendiğin satırı seç. Daha fazlasına ihtiyacın olursa, ilk duraklamayı anladıktan sonra bunları tek tek ekle.



Bir başka hata da kesme noktalarını kodda bırakmaktır. Program normal çalışması gerekirken durmaya devam eder, bu yüzden işin bittiğinde onları kaldır. Kesme noktaları hata ayıklama içindir, normal çalışma için değil. Onları kaldırmayı unutursan program beklenmedik şekilde duraklar ve bu can sıkıcı olabilir. Sorunu düzelttikten sonra kesme noktalarını her zaman temizle.



Kesme noktası, hata ayıklama sırasında bir programı seçilen satırda duraklatır. Sonra değişkenleri inceler ve adım adım çalıştırırsın, bu da hızlı bir programı kolayca gözlemlemeni sağlar. Bir seferde tek bir kesme noktası kullanmayı ve işin bittiğinde onları kaldırmayı unutma. Bu basit araç, çalışan bir programın içini görmene ve hataları verimli şekilde bulmana yardımcı olur.