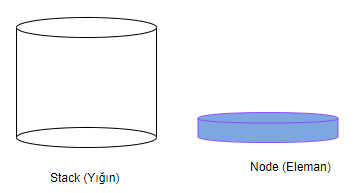
Yığın (Stack)

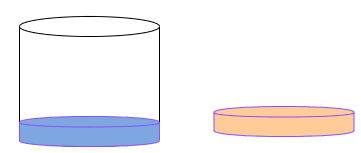
Türkçesi “Yığın” olan stack kelimesi bir şeyleri üst üste yığmaktan geliyor gibi düşünülebilir. Düz bir zemine birşeyler koymaya başlarsanız, en altta kalan ve onun üzerine gelenler şeklinde bir yığılma görürsünüz. Bu yığından bir eleman almaya kalktığınızda ise en üstten almanız gerekecektir. Diğer bütün veri yapıları gibi yığın da belli bir mantık ve amaçla, verileri tutmaya yarar. Özel kullanımlar dışında yığınlar genellikle program bloklarının tutulduğu yerlerdir. Yani iç içe çağrılan fonksiyonlarda, bir önceki fonksiyonun kaldığı yerin tutulması olarak düşünülebilir. Her bir fonksiyon çağrıldığında bir öncekinin kaldığı yeri yığına atar ve en içteki fonksiyon bittiğinde diğer adresleri sırayla yığından çekmeye başlar. Bu kalisk programlama yapısıdır ve hemen her yerde kullanılır. Zaman zaman heap ile karıştırılır fakat heap kuyruk (FIFO) yapısına sahipken stack (LIFO) yapısına sahiptir. Bu “son giren ilk çıkar”, “Last in First out” kuralıdır.

Stack’i silindir biçimindeki bir bardak olarak düşünelim. İçerisindeki elemanları da bardağa atılan aynı ağız genşiliğine sahip demir paralar olarak varsayalım. Bardaktan bir şey çıakrmak istediğimizde mutlaka en son attığımız parayı ilk çıkarmamız gerekecektir. Yeni bir para eklemek isteidğimizde ise en son atılanın üstüne konulması gerekecektir.

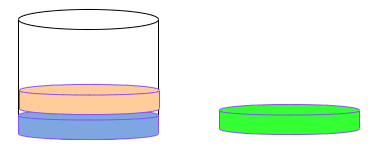
Bir de şekille anlatırsak:



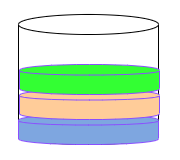
Boş bir yığına mavi node’u koymak isteyelim Push(mavi):



Şimdi de turuncu node ‘u koyalım Push(turuncu):

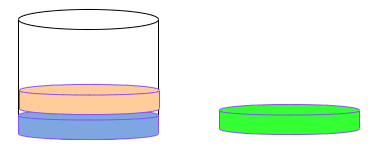


Yeşil node’u koymak istersek Push(yesil):



Şeklinde bir yerleşim tahayyül edebiliriz.

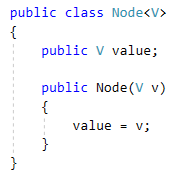
Bir şey çıkarmak istediğimizde, doğal olarak (yani bardak tasarımımıza göre) en üsttekini çıkarabileceğiz:



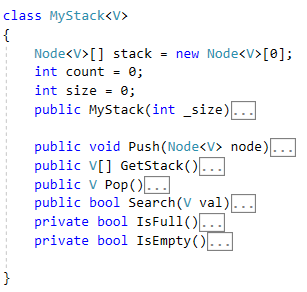
Sanırım şekilsel olarak stack anlaşıldı.

Gelelim bunu kod’da nasıl yapacağımıza:

C# dilinde her tip için uygun olabilecek bir “Node” sınıfı tasarlıyoruz, bu bizim paramıza tekabül ediyor.



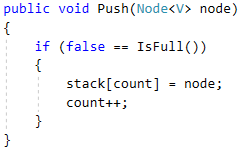
Daha sonra Yığın için düşündüğümüz bardak tasarımımızı yapıyoruz. Mantıken bir bardağı bu amaçla tasarlarken, bardağın kaç madeni para alacağını, ekleme ve çıkarma işlemleri sırasında neler yapması gerektiğini söylememiz gerekecektir. Burada da bu mantığı kullanıyoruz:



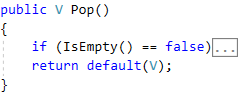
Stack’ler kullanım alanları itibari ile içerisindeki veriyi array olarak tutar. Biz de o amaçla “stack” adında array tanımladık. Size’ını MyStack objemizi tanımlarken vereceğiz. Push methodunu ekleme, Pop methodunu ise stack’ten veri almak için kullanacağız. Ayrıca tüm stack’in o anki fotoğrafı için GetStack, merak edilen bir elemanın orada olup olmadığını anlamak için de Search methodlarını kullanacağız.

Ekeleme ve çıkarma yaparken Stack’in durumunu control etmek için de private IsFull ve IsEmpty methodlarımız bulunmaktadır.

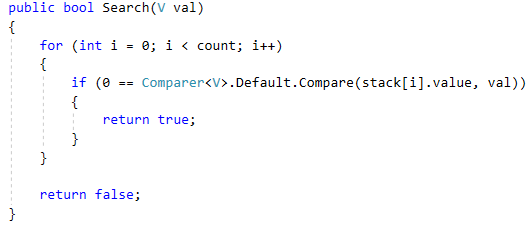
Push Methodu:



Pop mehtodu:



Search methodu:



Bu veri tipinin zaman karmaşıklığı (time complexity) ise söyledir:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pop | Push | Search |
| Zaman Karmaşıklığı  (Time Complexity) | O(1) | O(1) | O(n) |

Daha iyi anlaşılması ve çalışan halinin görülmesi için github kodunu indirebilirsiniz:

<https://github.com/nilaytufek/Stack>