Bağlı Listelere Dizilerle Yaklaşım

Canburak TÜMER
canburak@asestasarim.com

Giriş

C programlama dilinde aynı tipte birden çok ve birbiriyle ilişkili olan veriler genel olarak diziler aracılığıyla tutulmaktadır. Ancak dizilerde veri saklamayla ilgili bazı sıkıntılar olmuştur bunlar:

- Dizilere eklenen yeni elemanın dizi sıralı uygun yere değil dizinin sonuna eklenmesi, Örn: 3-5-7 diye giden bir diziye 4 eklenmek istenirse dizinin son hali 3-5-7-4 olur oysa dizinin sıralı olması için 3-4-5-7 olarak gitmesi gerekir.
- Sıralı olarak yazılmak istendiği takdirde var olan dizi elemanlarının ötelenmesi gerekliliği ve bu işlemin programa boyut ve süre olarak getirdiği ek yükler,
- Sırasız giden bir diziyi sıralamak için fonksiyon kullanımı zorunluluğu, Örn: Bubble Sort Algoritması
- Programın başında sabit olarak ayrılan bir bellek alanı dolayısıyla bellek kullanımında verimsizlik.

Sorunların çözülmesi amacıyla geliştirilen veri yapılarından biri olan bağlı veya bağlaçlı listeler bu sorunları şu şekilde aşmıştır:

- Sıralı ekleme fonksiyonu sayesinde yeni eklenecek elemanlar doğrudan listenin uygun konumuna kaydedilerek daha sonra sıralama veya eklerken öteleme külfetinden program kurtarılmış,
- malloc(), calloc() ve free() fonksiyonları sayesinde belleğin dinamik kullanımı sağlanmış ve bu sayede boş olan veri konumları bilgisayarın kullanımını için temizlenirken, gerekli görüldüğü takdirde boş bellek alanları programın kullanımına sunulmuş bu sayede belleğin verimli kullanılması garanti altına alınmıştır.

Bağlı listelerin kaynak olduğu temel sorunlara gelirsek:

- Deneyimsiz programcılar için bir muamma olan işaretçi kavramıyla ilgili çok da basit olmayan aritmetik ve mantıksal işlemlerin kullanım gerekliliği,
- (,) , & , * , . , -> gibi birden çok kimisi aynı işlevi gören operatörlerin kodlamada sözdizimi hatası olasılıklarını artırması,
- Yanlış işaretçi aritmetiği sonucu bellekte saklanan verilere tekrar ulaşılamaması veya başka yazılımlara ait verilere erişim ve verilerin üzerine yazma olasılığının varlığı,

Yukarıda sayılan avantaj ve dezavantajlar nedeniyle bir grup veriyi saklamak için veri yapısı olarak sadece dizileri veya sadece bağlı listeleri kullanarak tam verim alınamayacağı görülmektedir. Bu sebeple bu iki veri yapısının birleşiminden ortaya çıkacak olan bir veri yapısı verileri saklama, işleme ve erişme açısından daha avantajlı olacaktır. Bu veri yapısı hem deneyimsiz programcılar için de kolay olabilmeli hem de küçük, orta ve büyük boyutlu programlar için yeterince etkin olmalıdır. Bu durum da ortaya *bağlı listelere dizilerle yaklaşım metodu* çıkmaktadır.

Bağlı Diziler

Bahsi geçen metodu incelediğimiz takdirde ilgili verileri ve bunlardan sonra ve/veya önce gelen verileri tutan bir yapıya ihtiyacımız olduğunu göreceğiz. Bu ana kadar bağlı listelerden hiçbir farkı olmasa da veri yapısının asıl farklılıkları bu noktada devreye girmektedir. Yapıya ait bir dizi tanımlayarak veri yapısı dizisinin oluşumu sağlandıktan sonra *bağlı listelere dizilerle yaklaşım metodu* başlamış olur. Bu metotta yapının içindeki önceki ve sonraki verileri tutan işaretçilerin yerini tamsayı tipinde değişkenler alır. Bunun nedeni artık erişilmek istenen veri bellekte herhangi bir konumda değil bir dizide tutulmasıdır. Artık önceki ve sonraki değişkenleri ilgili elemanın dizi indisini saklamaktadır.

Bu veri yapısının bize sunduğu avantajlara gelirsek:

- Veriler bağlı listelerde olduğu gibi belleğin rasgele noktalarında bulunmayıp bir dizide, bir arada tutulduğu için kaybolma imkânları yok.
- İşaretçi aritmetiği kullanılmadığı için bellekte yanlış konumlara gidip yanlış verilerin üzerine yazma veya yanlış verileri okuma gibi bir problemin çıkma ihtimali azaltılmış olur.
- Veriler arasındaki bağ bir şekilde kopsa dahi gerekli veri dizi baştan sona taranarak bulunabilir.
- Sıralama algoritması, sıralamak için öteleme gibi işlemlere gerek duymadan doğrudan bağlı listelere sıralı ekleme fonksiyonuna yakın bir fonksiyon kullanarak sıralı ekleme ve silme yapabilme imkanı sunar.
- * , & , . , -> gibi çok sayıda operatör kullanımına ihtiyaç olmadığından mantık ve sözdizimi hatalarının büyük ölçüde önüne geçilir.

Dezavantajları ise:

- Yazılım çalıştığında gerekli bellek miktarı ayrılacağı için dinamik bellek kullanımı yok.
- Veri yapısı bir dizi olduğu için azami eleman sayısı belli aşılamaz bu nedenle bağlı listeler kadar esnek değil.
- Silinen eleman belleğe geri kazandırılamıyor sadece mantıksal olarak temizleniyor.

Son olarak bu veri yapısıyla ilgili ayrıntılara girersek, bu veri yapısı *a) Tek bağlı diziler, b) Çift bağlı diziler* olarak ikiye ayrılmaktadır. Dizi elemanlarının bağlanma mantığı bağlı listelerden farksızken, elemanların bellekte saklanma şekilleri dizilerler aynıdır. Bu veri yapısında bağlı listelerdeki "head node" benzeri bir ilk değer tutan eleman bulunası gereklidir. Aksi takdirde sıralı ekleme işleminde bir süre sonra problemlerin ortaya çıkma ihtimali mevcuttur. Bu nedenle dizinin 0 indisli elemanı bu veri yapısının "head node" u olarak ayarlanmalı ve önceki veri değeri "NULL", sonraki veri değeri ise ilk dolu verinin indisi olarak ayarlanmalı ve böylece her durumda ilk veriye ulaşım sağlanmalıdır. Bunların yanı sıra *bağlı diziler* dairesel olarak da yapılabilir bunun için son elemanın "head node" un gösterdiği elemanı göstermesi yeterli olacaktır.

Görsel Öğeler Yardımıyla *Bağlı Diziler*

Aşağıda ilgili veri yapısını örnekleyen şemalar mevcuttur.

Tek Bağlı Diziler

10 elemanlı bir tek bağlı dizi için şema:

İndis	Veri	Sonraki
0	NULL	6
1	2	4
2	8	NULL
3	5	7
4	3	5
5	4	3
6	1	1
7	6	9
8	NULL	NULL
9	7	2

Çift Bağlı Diziler

5 elemanlı bir çift bağlı liste için şema:

İndis	Veri	Önceki	Sonraki
0	NULL	NULL	2
1	NULL	NULL	NULL
2	1	NULL	4
3	3	4	NULL
4	2	1	3