

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ

Veri Yapıları ve Algoritmalar Dönem Ödevi

Furkan Kuzucuk / 2204040151

ÖN SÖZ

Bu ödevde, **Çift Bağlı (Hybrid) Dairesel Liste** veri yapısının C veya C++ dilinde nasıl implement edileceği gösterilecektir. Çift bağlı dairesel liste, her bir elemanının hem bir önceki hem de bir sonraki elemana işaret ettiği bir liste türüdür. Bu veri yapısı, elemanlar arasında ileri ve geri gitmeye olanak tanır ve son elemandan başa doğru dönebilme yeteneği sağlar.

**İÇİNDEKİLER**

[ÖN SÖZ ii](#_Toc197907297)

[Giriş 1](#_Toc197907298)

[Kullanılan Dil ve Kütüphaneler 1](#_Toc197907299)

[Kullanılan Argümanlar 2](#_Toc197907300)

[PSEUDOCODE 3](#_Toc197907301)

[Akış Diyagramı 4](#_Toc197907302)

[C Kodu 6](#_Toc197907303)

[Program Çıktısı 7](#_Toc197907304)

Giriş

Veri yapıları, yazılım mühendisliğinde önemli bir rol oynar ve veri işleme süreçlerini daha verimli hale getirebilmek için çeşitli türlerde veri yapıları kullanılır. Bu yapılar, veri manipülasyonu, ekleme, silme ve erişim işlemleri için optimize edilmiştir. Çift bağlı (doubly linked) liste, bu veri yapılarından biridir ve her bir elemanın, bir sonraki ve bir önceki elemanı işaret ettiği bir yapıyı ifade eder. Ancak, her iki yönlü bağlantıların kullanıldığı geleneksel çift bağlı liste yapısının yanı sıra, çift bağlı dairesel liste (hybrid doubly circular linked list) veri yapısı da farklı özellikler sunar.

Çift bağlı dairesel liste, hem ileriye hem de geriye doğru gezinmeyi sağlayan bir yapıdır. Ayrıca, son düğüm ile ilk düğüm arasında bir bağlantı bulunduğundan, bu yapı, son elemandan ilk elemana veya ilk elemana son elemana geçiş yapabilme özelliği sunar. Bu, özellikle döngüsel yapılara ihtiyaç duyulan uygulamalarda faydalı olabilir. Örneğin, bir müzik çalar uygulamasında şarkıların sürekli çalması ya da bir oyun simülasyonunda dairesel sıralama ihtiyaçları gibi durumlarda kullanılabilir.

Bu projede, C dilinde bir Çift Bağlı Dairesel Liste veri yapısının temel işlevlerini gerçekleştirerek, bu yapının nasıl çalıştığını ve nasıl uygulama geliştirilmesinde kullanılabileceğini göstereceğiz. Program, çift bağlı dairesel listeye eleman eklemek, elemanları sıralı olarak yazdırmak, listeyi tersten yazdırmak ve temel liste manipülasyonları yapmak için gerekli işlevleri içerecektir.

Bu programın amacı, kullanıcıya çift bağlı dairesel liste yapısının çalışma mantığını ve kodla nasıl implement edilebileceğini anlamasında yardımcı olmaktır. Bu veri yapısının, özellikle ileri ve geri yönlü gezinme ve dairesel yapının sağladığı avantajlar üzerinde durulacaktır.

Kullanılan Dil ve Kütüphaneler

* Dil: C
* **Kütüphaneler:**
* stdio.h: Giriş/çıkış işlemleri için kullanılır. Kullanıcıdan veri alırken ve ekrana yazdırırken kullanılır.
* stdlib.h: Bellek yönetimi için kullanılır. Dinamik bellek tahsisi (malloc, free) gibi işlemleri gerçekleştirir.

Kullanılan Argümanlar

1. struct Node\* head (Ana işaretçi):

Açıklama: Bu argüman, listeyi temsil eden baş işaretçidir. Çift bağlı dairesel listede, tüm işlemler bu işaretçi üzerinden yapılır. Eğer liste boşsa, head işaretçisi NULL olur.

Kullanım: Listeyi başlatırken ya da eleman eklerken bu işaretçi kullanılır.

2. struct Node\* newNode:

Açıklama: Bu argüman, yeni bir düğüm oluşturmak için kullanılır. newNode, yeni bir liste elemanı eklemek amacıyla dinamik olarak tahsis edilen bir düğümdür.

Kullanım: createNode fonksiyonu, yeni düğüm oluşturduğunda newNode işaretçisi kullanılır. Ayrıca, eleman eklerken bu işaretçi ile listeye eklenmesi gereken düğüm gösterilir.

3. int data:

Açıklama: Bu, düğümün taşıyacağı veriyi ifade eder. Her düğüm bir int veri türü içerir.

Kullanım: Eleman eklerken, kullanıcıdan alınan veriyi temsil eder. createNode fonksiyonuna parametre olarak verilerek her düğüme veri eklenir.

4. struct Node\*\* head (Pointer to pointer):

Açıklama: Bu, baş işaretçinin kendisini değiştirmek için kullanılır. C dilinde işaretçileri değiştirmek için bir işaretçinin işaretçisine ihtiyaç duyulur.

Kullanım: insertEnd fonksiyonu içinde kullanılır. Baş işaretçisini değiştirmek gerektiğinde (head = newNode), bu şekilde yapılır.

5. struct Node\* temp:

Açıklama: Bu geçici işaretçi, liste üzerinde gezinirken kullanılır. Listeyi baştan sona ya da sondan başa doğru yazdırmak için kullanılır.

Kullanım: printList ve printReverse fonksiyonlarında, listede ilerlerken geçici bir işaretçi olarak kullanılır. İleri veya geri yönlü gezinmeyi sağlar.

6. \*head:

Açıklama: head işaretçisi bir pointer to pointer olduğundan, burada \*head kullanılarak baş işaretçisinin kendisine erişilir.

Kullanım: insertEnd fonksiyonu içinde head işaretçisinin değerini değiştirmek için kullanılır. Listeye yeni bir eleman eklerken baş işaretçisini günceller.

7. next ve prev:

Açıklama: Bu, her düğümün bir sonraki (next) ve önceki (prev) düğüme işaret eden işaretçileridir.

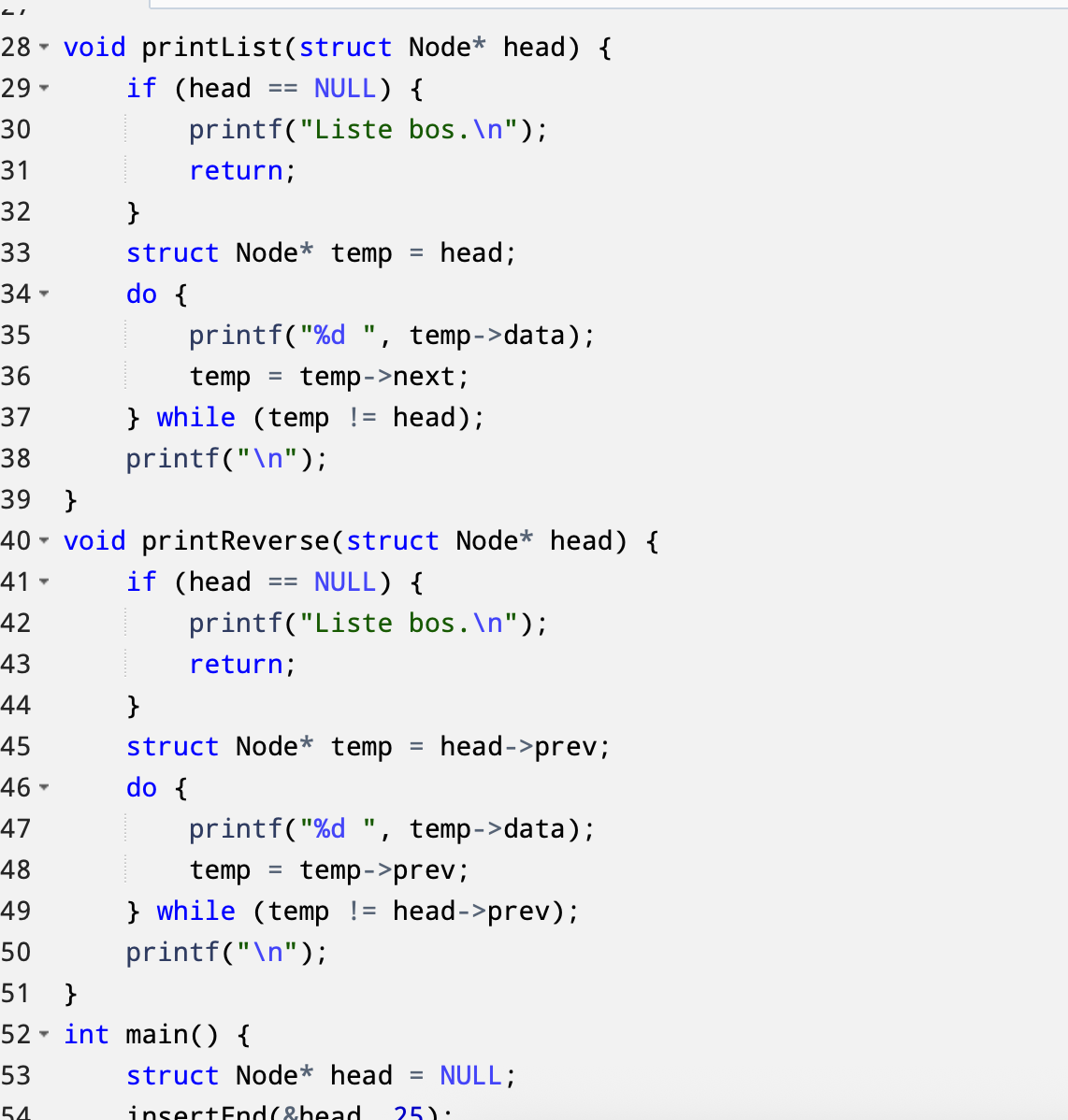
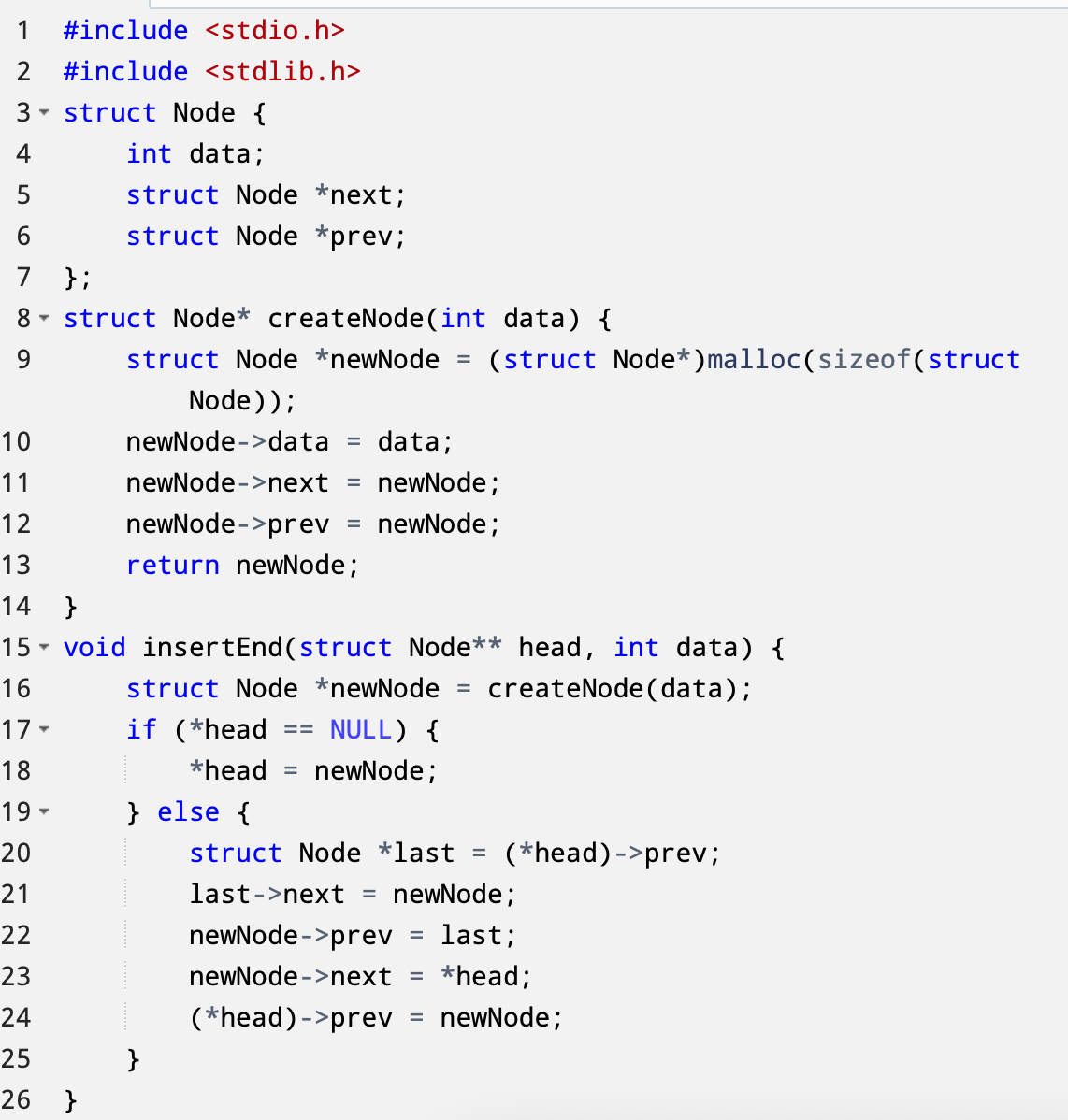
Kullanım: Listeyi dairesel hale getirebilmek için next ve prev işaretçileri güncellenir. Listeyi baştan sona ya da sondan başa yazdırırken bu işaretçiler kullanılır.

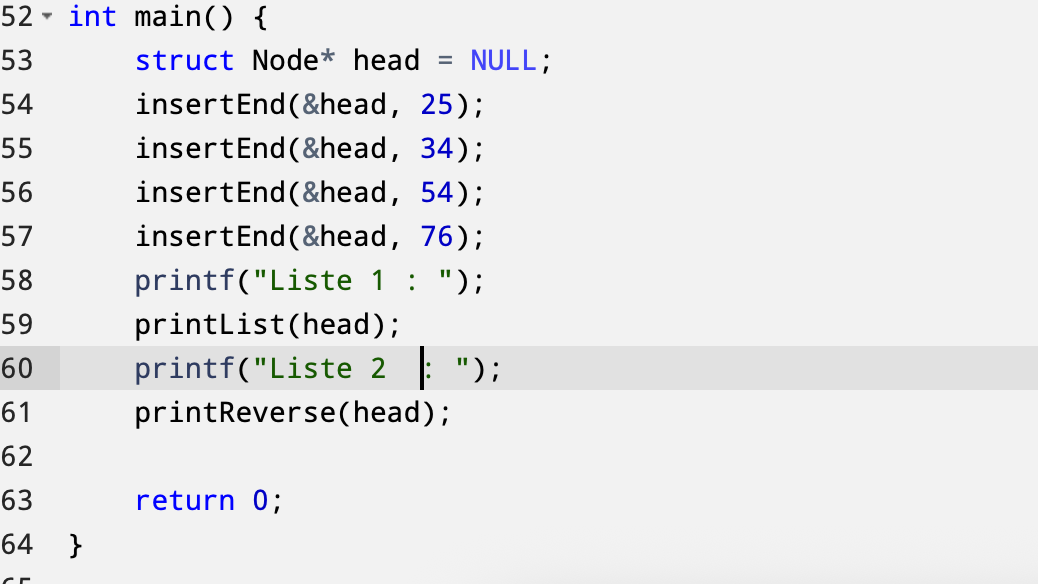
PSEUDOCODE

1. Başlat
2. Listeyi başlat: head = NULL
3. Node yapısını tanımla: struct Node { int data; struct Node \*next; struct Node \*prev; };
4. Yeni bir düğüm ekle:
   1. Yeni düğüm oluştur.
   2. Liste boşsa, head'i bu düğüme işaret et.
   3. Liste boş değilse, son düğümün next'ine yeni düğüm ekle ve son düğümü yeni düğüm olarak güncelle.
5. Eleman silme:
   1. Silinecek elemanı bul.
   2. Önceki ve sonraki düğümlerin birbirine işaret etmesini sağla.
   3. Belleği serbest bırak.
6. Listeyi yazdır:
   1. Baştan sona ve sondan başa doğru listeyi yazdır.
7. Sonlandır.

Akış Diyagramı

* **Başlangıç**
* **Listeyi Başlat**:
* head = NULL (Listeyi başlat)
* **Yeni Eleman Ekleme**
* Eğer liste boşsa, yeni düğümü head olarak ayarla.
* Eğer liste boş değilse, yeni düğümü son elemana ekle ve uygun yönlendirmeleri yap.
* **Eleman Silme**
* Silinecek elemanı bul ve uygun yönlendirmeleri yap.
* **Listeyi Görüntüle**
* Listeyi baştan sona ve sondan başa doğru yazdır.
* **Sonlandır**

C Kodu



Program Çıktısı

