Kuyruk

Kuyruk veri yapısı çıkın veri yapısına cok benzer.Çıkın yapısında olduğu gibi,bu yapıda elemanlar bir dizi halinde tutulur.Aynı şekilde kuyruk veri yapısında da iki tane işlem tanımlıdır : Veri ekleme ve veri silme. Bu işlemlerden veri ekleme,çıkında olduğu gibi listenin sonuna eklenır.Fakat bir veri silineceği zaman ,listenin son elemanı değil,listenin ilk elemanı silinir.Bu ise,kuyruk veri yapısının ilk giren ilk çıkar tarzı bir veri yapısı olduğunu gösterir.Aynı kuyrukta bekleyen insanlar gibi.

Sabit Dizi ile Kuyruk Tanımı

Veriler dizi değişkeninde tutulur.bas değişkeni kuyrugun ilk elemanını göstermektedir.Son değişkeni kuyrugun son değşkenini ifade etmketedir.Kuyruk ilk yaratıldığında bas ve son değişkenleri birbirine eşit ve 0 değerini almaktadırlar.Kuyrygun elemanları dizide mutlaka sıfırıncı yerden başlamak zorunda değildirler.Kuyruk ,dizinin ortasından başlamıs ve baskka bir yerinde bitmiş olabilir.

Tam sayılar içeren sabit dizi ile bir kuyruk uygulaması

```
1 public class Kuyruk{
2 Ornek dizi [];
3 int bas;
4 int son;
5 int N;
6 public Kuyruk(int N){
7 dizi = new Ornek[N];
8 this.N = N;
9 bas = 0;
10 son = 0;
11 }
12 boolean kuyrukDolu(){
13 if (bas == (son + 1) % N)
14 return true;
15 else
16 return false;
17 }
18 boolean kuyrukBos(){
19 if (bas == son)
20 return true;
21 else
22 return false;
23 }
24 }
```

Verilen bir kuyrugun dolu olup olmadığını döndüren fonksiyon kuyruk_dolu adıyla verilmiştir.Kuyrugun en fazla N-1 eleman aldığından kuyrugun sonunu gösteren işaretci kuyrugun başını gösteren işaretciden bir önde olduğunda kuyruk dolu olacaktır.Verilen bir kuyrugun boş olup olmadığını döndüren fonksiyon kuyruk_bos adıyla verilmiştir.Kuyrugun bas işaretcisi ile son işaretcisi aynı yeri gösteriyorsa kuyruk boş olacaktır.

Sabit dizi ile uygulanan bir kuyruga yeni bir eleman ekleyen algoritma

```
1 void kuyrugaEkle(Ornek yeni){
2 if (!kuyrukDolu()){
3 dizi [son] = yeni;
4 son = (son + 1) % N;
5 }
6 }
```

Eğer bas değişkeni son değişkeninden bir eksik ise kuyruk dolmuş demektir(2).Bu durum söz konusu değilse,kuyrugun son işaretcisi ile gösterilen yere yeni eleman yerleştirilir(3).Eğer son değişkeni diznin sonunda değilse bir artırılır,sonubda ise bir artırıldıgında dizinin dışına çıkacagı için dizinin ilk elemanını göstermesi sağlanır(4).

Sabit dizi ile uygulanmıs bir kuyruktan eleman silen ve o elemanı döndüren algoritma

```
1 Ornek kuyrukSil(){
2 Ornek sonuc;
3 if (!kuyrukBos()){
4 sonuc = dizi[bas];
5 bas = (bas + 1) % N;
6 return sonuc;
7 }
8 return null;
9 }
```

Kuyruktan bir eleman silinmek istendiğinde önce kuyrugun boş olup olmadığı kontrol edilmelidir.bas ve son değişkebnleri birbirine eşitse kuyruk boştur(3).Eğer kuyruk boş değilse kuyrugun ilk elemanı döndürülür(4,6) ve kuyrugun ilk elemanını gösteren bas işaretcisi bir artırılır.Fakat eğer bas değişkeni dizinin son elemanını gösteriyorsa değeri bir artırldıgında dizinin dışına çıkacaktır.Bu durumda yapılması gereken bu sefer de bas değişkeninin dizinin ilk elemanını göstermesını sağlamaktır(5).

Bağlı Liste ile Kuyruk Tanımı

Bağlı liste ile luyruk yaratıldığında bas ve son değişkenleri NULL ilk değerine eşitlenir.

Tam sayılar içeren baglı liste ile bir kuyruk uygulaması

```
1 public class Kuyruk{
2 Eleman bas;
3 Eleman son;
4 public Kuyruk(){
5 bas = null;
6 son = null;
7 }
8 boolean kuyrukBos(){
9 if (bas == NULL)
10 return true;
11 else
12 return false;
13 }
14 }
```

Kuyruga Eleman Ekleme

```
1 void kuyrugaEkle(Eleman yeni){
2 if (!kuyrukBos())
3 son. ileri = yeni;
4 else
5 bas = yeni;
6 son = yeni;
7 }
```

Bağlı liste yapısı ile kuyruk yapısı uygulanmışsa kuyrugun dolmuş olması söz konusu olamaz,çünkü bağlı listelerde hafizada yer olduğu müddetçe,hafızada yer açılıp eleman eklenir.

Kuyruktan Eleman Silme

```
1 Eleman kuyrukSil(){
2 Eleman sonuc;
3 sonuc = bas;
4 if (!kuyrukBos()){
5 bas = bas. ileri ;
6 if (bas == null)
7 son = null;
8 }
9 return sonuc;
10 }
```

Bağlı liste ile uygulanmış kuyruk yapısında ilk elemanı silmek için yapılacak işlem sadece ilk elemanı gösteren işaretciyi bir ilerletmektir(5).