KUYRUK VERİ YAPISI

Kuyruk yapısı ilk giren ilk çıkar mantığıyla çalışan bir yapıdır. (Yazıcı kuyruğu gibi)

1) Dizi üzerinde kaydırmalı Kuyruk Yapısı

Son pozisyona eklenir

10	**80**	**40**	**70**	

İlk pozisyondan çıkar

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

#define KBoyut 100 // kuyruk boyutu tanımlanmaktadır.
typedef struct Kuyruk
{     int son; // kuyruğun son elemanının indisini göstermektedir
      int eleman[KBoyut];
}Kuyruklar;

Kuyruklar Yeni_Kuyruk;
Yeni_Kuyruk.son=-1; // başlangıçta kuyruk boştur

int Kuyruk_Dolumu() //kuyruk dolu ise -1, kuyrukta eklemek için yer varsa 1 dönmektedir
{     if (Yeni_Kuyruk.son>=KBoyut-1) return -1;else return 1;
}

int Kuyruk_Bosmu() //kuyruk boş ise -1, kuyruktan çıkacak eleman varsa 1 dönmektedir
{     if (Yeni_Kuyruk.son==-1) return -1;else return 1;
}

int Kuyruga_Ekle(int sayi)
{   if (Kuyruk_Dolumu()==-1)
      {     printf("Kuyruk doludur. Ekleme yapılamaz\n");
            return -1; // ekleme başarısız
      }
      else
      {     Yeni_Kuyruk.son++; // kuyruğun sonunu gösteren indis 1 arttırılır
            Yeni_Kuyruk.eleman[Yeni_Kuyruk.son]=sayi;
            return 1; // ekleme başarılı
      }
}

int Kuyruktan_Cikar()
{     char cikan_eleman; int i;
      if (Kuyruk_Bosmu()==-1)
      {     printf("Kuyruk Bos. Çıkarma yapılamaz\n");
            return -1;
      }
      else
      {     cikan_eleman=Yeni_Kuyruk.eleman[0]; // kuyruğun ilk elemanı çıkarılır
            for(i=1;i<=Yeni_Kuyruk.son;i++) // kaydırma işlemi yapılır
                  Yeni_Kuyruk.eleman[i-1]=Yeni_Kuyruk.eleman[i];
            Yeni_Kuyruk.son--; // kuyruğun sonunu gösteren indis 1 azaltılır
            return cikan_eleman;
      }
}

void Listele()
{int i;
      for (i=0;i<=Yeni_Kuyruk.son;i++)
            printf("\n: %d",Yeni_Kuyruk.eleman[i]);
}
```

```
void main()
{
      int i;
      char secim;
      int numara;
      Yeni_Kuyruk.son=-1;
      clrscr();
      while(1==1)
      {     clrscr();
            puts("\nEkleme\nCikarma\n\Listeleme\nCikis\nSecim?");
            secim=getchar();
            switch(secim)
            {     case 'e':
                        puts("Numarayi giriniz");
                        scanf("%d",&numara);
                         Kuyruga_Ekle(numara);
                         break;
                  case 's':printf("%d",Kuyruktan_Cikar());
                         break;
                  case 'l':
                         Listele();
                         getch();
                         break;
                  case 'c':
                         exit(0);
            }
      }
}
```

Dairesel Kuyruk

Dizi üzerinde kaydırma gereksiz yere her çıkarma işleminde kuyruktaki veri sayısından bir eksik kaydırma işlemine gerek duymaktadır. Bu nedenle çok fazla sayıda eleman bulunan kuyruklarda kaydırma işlemi uzun sürmektedir. Dairesel kuyruk yapısında dizinin son elemanının bir sonraki elamanı dizinin ilk elemanıdır. Böylece kuyrukta bir halka varmış gibi hareket edilebilir. Kaydırma işlemi yerine kuyruktaki ilk elamanı gösteren ikinci bir değişken (bas) kullanılır. Kuyruktan alma işlemi bu değişkenin gösterdiği gözden yapılır.

Örnek:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#define KBoyut 100   // kuyruk boyutu tanımlanıyor
typedef struct Kuyruk
{     int son; // eleman eklemek için kullanılır. Eklemede 1 arttırılır
      int bas; // eleman çıkarmak için kullanılır. Çıkarmada 1 arttırılır
      int sayi; // kuyruktaki eleman sayısını tutan değişkendir
      int eleman[KBoyut];
}Kuyruklar;

Kuyruklar Yeni_Kuyruk;
Yeni_Kuyruk.son=-1; // başlangıçta kuyruk boştur
Yeni_Kuyruk.bas=-1; // başlangıçta kuyruk boştur
Yeni_Kuyruk.sayi=0; // başlangıçta kuyruk boştur

int Kuyruk_Dolumu() //kuyruk dolu ise -1, kuyrukta eklemek için yer varsa 1 dönmektedir
{     if (Yeni_Kuyruk.sayi==KBoyut) return -1;else return 1;
}

int Kuyruk_Bosmu() //kuyruk boş ise -1, kuyruktan çıkacak eleman varsa 1 dönmektedir
{     if (Yeni_Kuyruk.sayi==0) return -1;else return 1;
}

int Kuyruga_Ekle(int numara) // kuyruğa ekleme işleminde son değişkeni 1 arttırılır
{     if (Kuyruk_Dolumu()==-1)
      {     printf("Kuyruk doludur. Ekleme yapılamaz \n");
            return -1; // Ekleme başarısız
      }
      else // gerekirse dizinin son hücresinden ilk hücresine geçiş yapılmaktadır
      {     Yeni_Kuyruk.son=(Yeni_Kuyruk.son+1)%KBoyut;
            Yeni_Kuyruk.eleman[Yeni_Kuyruk.son]=numara;
            Yeni_Kuyruk.sayi++; // eleman sayısı 1 arttırılır
            return 1; // Ekleme başarılı
      }
}

int Kuyruktan_Cikar() // kuyruktan çıkarma işleminde bas değişkeni 1 arttırılır
{int cikan_eleman; int i;
      if (Kuyruk_Bosmu()==-1)
      {     printf("Kuyruk Bos. Çıkarma yapılamaz \n");
            return -1;
      }
      else // gerekirse dizinin son hücresinden ilk hücresine geçiş yapılmaktadır
      {     Yeni_Kuyruk.bas=(Yeni_Kuyruk.bas+1)%KBoyut;
            cikan_eleman=Yeni_Kuyruk.eleman[Yeni_Kuyruk.bas];
            Yeni_Kuyruk.sayi--; // eleman sayısı 1 azaltılır
            return cikan_eleman;
            }
}
```

```
void Listele()
{     int i;
      for (i=1;i<=Yeni_Kuyruk.sayi;i++)
            printf("\n: %d",Yeni_Kuyruk.eleman[(Yeni_Kuyruk.bas+i)%KBoyut]);
}

void main()
{     int i;
      char secim;
      int numara;
      clrscr();
      while(1==1)
      {     clrscr();
            puts("\nEkleme\nCikarma\n\Listeleme\nCikis\nSecim?");
            secim=getchar();
            switch(secim)
            {     case 'e':
                        puts("Numarayi giriniz");
                        scanf("%d",&numara);
                        Kuyruga_Ekle(numara);
                        break;
                  case 's':Kuyruktan_Cikar();
                        break;
                  case 'l':    Listele();
                        getch();
                        break;
                  case 'c':   exit(0);
            }
      }
}
```