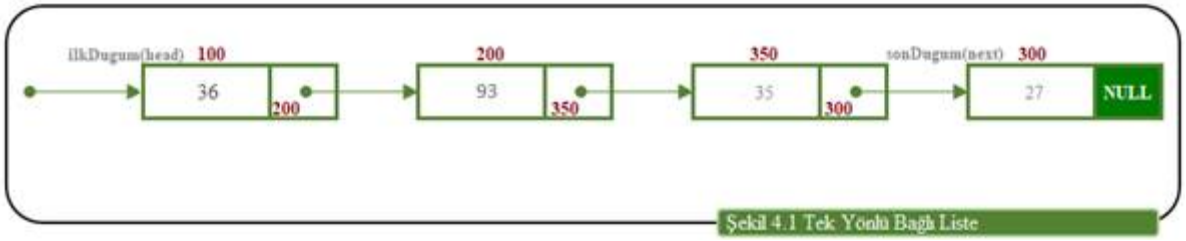
4. DAİRESEL BAĞLI LİSTELER

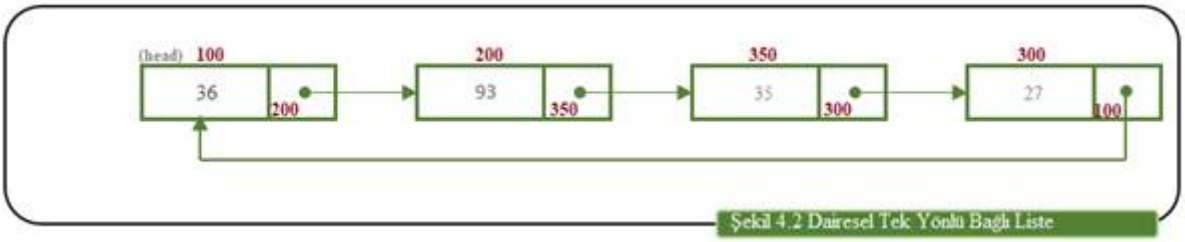
Giriş

Bu kitabın ikinci bölümünde açıklandığı gibi, tek yönlü bağlı listelerin son düğümünün *sonraki* işaretçisi her zaman *NULL* değerini gösterir. Tekrar hatırlatmak amacıyla, tek yönlü bağlı listenin bellekteki yerleşimi şekil 4.1 'de sembolik olarak gösterilmiştir.



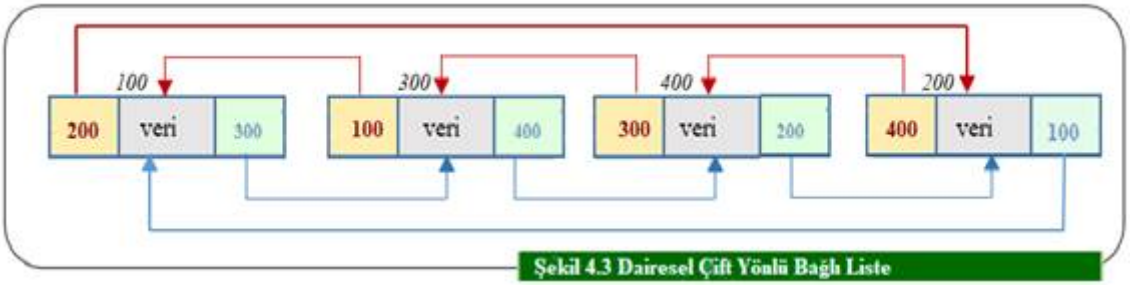
Şekil 4.1 Tek Yönlü Bağlı Liste

Tek yönlü bağlı liste yapısına ilave bir özellik eklenerek, listenin en son elemanının *sonraki* (next) işaretçisinin listenin ilk elemanını göstermesi sağlanabilir. Başka bir ifade ile tek yönlü bağlı listenin son elemanında, listenin başındaki elemanı gösterecek bir işaretçi bulundurularak oluşturulan listelere *Dairesel Tek Yönlü Bağlı Listeler* denir. Şekil 4.2'de dairesel tek yönlü bağlı liste gösterilmiştir.



Şekil 4.2 Dairesel Tek Yönlü Bağlı Liste

Benzer şekilde bir çift bağlantılı listede, listenin son elemanının sonraki *(next)* işaretçisi ile listenin ilk düğümünün *(head)* işaret edilmesi ve listenin ilk düğümünün önceki *(prev)* işaretçisi ile listenin son düğümünün işaret edilmesi ile oluşturulan listelere *Dairesel Çift Yönlü Bağlı Listeler* denir. Şekil 4.3'te dairesel çift bağlı bir liste gösterilmiştir.

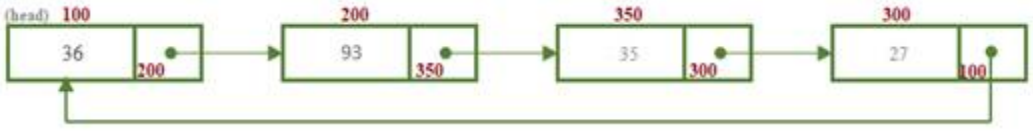


Şekil 4.3 Dairesel Çift Yönlü Bağlı Liste

Bu bölümde dairesel tek yönlü bağlı listeler ile dairesel çift yönlü bağlı listeleri ele alacağız. Her iki dairesel bağlı liste için konu başlıklarımız dairesel bağlı listelerin oluşturulması, Dairesel bağlı listelerin sonuna, başına listede bulunan elemanların arasına yeni eleman eklemek, dairesel bağlı listelerden eleman silmek, dairesel bağlı listelerde arama yapmak şeklinde olacaktır. Dairesel çift bağlı listelerde arama yapmak ve listeden eleman silmek konuları sıra sizde çalışması olarak öğrencilere bırakılmıştır.

4.1. Dairesel Tek Yönlü Bağlı Listeler

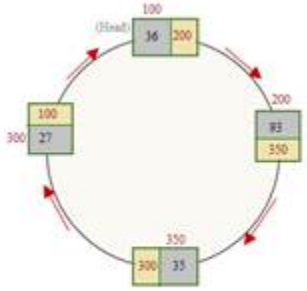
Dairesel tek yönlü bağlı bir listelerin, doğrusal olan tek yönlü bağlı listelerden farkı, listenin herhangi bir düğümünden listenin diğer düğümlerine ulaşılmasının mümkün olmasıdır. Bu yapıda listenin elemanları arasında traverse (dolaşma) işlemi, listenin herhangi bir elemanından başlanıp, başlangıç noktasına kadar elemanların tamamı dolaşılarak gerçekleştirilir. Aşağıda dairesel tek yönlü bağlı liste tekrar gösterilmiştir.



Şekilden de görüleceği gibi, dairesel tek yönlü bağlı listelerde son düğümün sonraki işaretçisi listenin ilk düğümüne **(head)** işaret eder. Hatırlanacağı gibi dairesel olmayan tek yönlü bağlı listede son düğümün işaretçisi daima **NULL** değerini alıyordu, liste elemanları arasında her zaman ilk düğümden başlanıp son düğüme kadar dolaşılabiliyordu ve dolaşma işlemi son düğümden sonra tekrar ilk düğüme dönülerek yapılamıyordu. Dairesel tek yönlü bağlı listelerde ise dolaşma işlemi son düğümden sonra ilk düğüme geçilerek sürdürülebilmektedir.

Yukarıda verilen şekil incelediğinde, ilk düğümün işaretçisinin değerinin 200 olduğu, 200'ün ilk düğümden sonra gelen düğümün adresi olduğu ve ilk düğüm listede ikinci sırada yer alan düğüme işaret ettiği görülmektedir. İkinci düğümün işaretçisinin değeri ise 350'dir ve 350 üçüncü düğümün adresidir. Dolayısı ile ikinci düğüm üçüncü düğüme, aynı şekilde üçüncü düğümün son düğüme işaret ettiği şekilden kolayca anlaşılmaktadır. Buraya kadar ilginç olan yeni bir durum yoktur. Çünkü birinci bölümde anlatılan tek yönlü bağlı listelerin yapısı da bu şekildeydi. Burada tek yönlü bağlı listelerden farklı olan durum; son düğümün işaretçisinin değerinin **NULL** yerine 100 olması ve ilk düğüme işaret etmesidir.

Ayrıca, burada dairesel tek yönlü bağlı listelerle ilgili belirtilmesi gereken önemli bir nokta var. Bu önemli nokta; dairesel tek yönlü bağlı listelerde son düğüm bulunmamasıdır. Çünkü dairesel tek yönlü bağlı listeler sonlanmaz. *Şekil 4.4*'te dairesel bağlı listenin mantıksal gösterimi verilmiştir.



Şekil 4.4. Dairesel Tek Yönlü Bağlı Liste

Aşağıda, dairesel tek yönlü bağlı listenin veri yapısı açıklanmıştır. Dairesel tek yönlü bağlı listeler için verilen bu yapıdan da görülebileceği gibi, yapının tanımı genel anlamda birinci bölümde verilen tek yönlü bağlı liste tanımıyla aynıdır. Bunun nedeni dairesel tek yönlü bağlı listelerin tek yönlü bağlı listelerden farkı sadece son düğümün işaretçisinin NULL olması yerine ilk düğüme (head) işaret eden bir değer almasıdır.

```
struct dugum {
    int veri;                   // Data
    struct dugum *sonraki;      // Adres
};
```

Dairesel Tek Yönlü Bağlı Listelerin Veri Yapısı

4.1.1. Dairesel Tek Yönlü Bağlı Liste Oluşturmak ve Listeye Eleman Eklemek

Dairesel tek yönlü bağlı liste oluşturmak için yazılan Program 4.1, genel itibariyle tek yönlü bağlı liste oluşturmak için kullanılan programa birçok açıdan çok benzemektedir. Tek yönlü bağlı liste oluşturan program, incelemeniz için bu bölümün sonunda Ek 1'de verilmiştir. Ek 1 'de verilen programı inceleyerek dairesel tek yönlü bağlı liste oluşturan program 4.1 ile bu programın arasındaki benzerlik ve farkları görebilirsiniz.

Daha önce de belirtildiği gibi dairesel tek yönlü bağlı liste yapısının (listede yer alacak olan düğümün yapısı) tanımı, ikinci bölümde anlatılan tek yönlü bağlı liste yapısının tanımı ile aynıdır. Program 4.1'de bu amaçla yazılan ifadeler aşağıda verilmiştir.

```
struct dugum{
 int veri;
 struct dugum *sonraki;
};
```

Dairesel tek yönlü bağlı listede bir düğümün yapısı aşağıda gösterilmiştir.

Yukarıdaki gösterilen düğüm yapasının, ikinci bölümde verilen düğüm yapısı ile aynı *(Bkz. Bölüm 2, Şekil 2.1)* olduğuna dikkat ediniz.

Program 4.1'in 8. Satırında, sonradan kullanılmak üzere global olarak yapı tipinde üç değişken tanımlanmıştır.

```
8   struct dugum *ilkDugum=NULL,*yeniDugum,*p;
```

Aynı şekilde *main()* fonksiyonu içerisinde 24. Satırda aşağıdaki yerel değişkenler tanımlanmıştır.

```
24      int i=0,n,s;
```

Program 4.1'de 25-29. Satırlar arasında listenin düğüm sayısının ve düğümün verisinin klavyeden girişi sağlanmıştır.

```
25      printf("\nListeye Kaç Dugüm Girilecek : ");
26      scanf("%d",&n);
27      printf("\n");
28      printf("%d. Düğümün          Verisi    : ",i);
29      scanf("%d",&s);
```

```
30      yeniDugum=(struct dugum*)malloc (sizeof(struct dugum));
```

30. satırda sizeof fonksiyonundan yararlanılarak bellekte yeniDugum için yer ayrılmıştır.

```
30      yeniDugum=(struct dugum*)malloc (sizeof(struct dugum));
```

31. satırda klavyeden girilen *s* değeri *yeniDugum* 'ün *veri* değişkenine atanmış ve 32. Satırda *yeniDugum* 'ün *sonraki* işaretçisi *NULL* yapılmıştır. 33 ve 34. Satırlarda yeniDugum ilkDugum ve p değişkenlerine atanmıştır. Bu aşamada *yeniDugum, ilkDugum* ve *p* listede oluşan düğümün üç ayrı ismidir.

```
31      yeniDugum->veri=s;
32      yeniDugum->sonraki=NULL;
33      ilkDugum=yeniDugum;
34      p=yeniDugum;
```

Aşağıda, bu aşamada dairesel tek yönlü bağlı listenin bellekteki durumu gösterilmiştir.

35 – 43. Satırlar arasında *for döngüsü* içerisinde aşağıdaki işlemler gerçekleşmiştir:

```
35      for(i=1;i<n;i++){
36         printf("%d. Düğümün       Verisi       : ",i);
37         scanf("%d",&s);
38         yeniDugum=(struct dugum*)malloc(sizeof(struct dugum));
39         yeniDugum->veri=s;
40         yeniDugum->sonraki=NULL;
41         p->sonraki=yeniDugum;
42         p=p->sonraki;
43      }
```

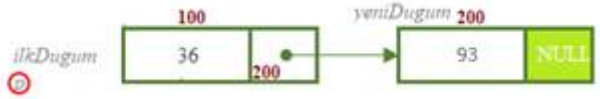
1. 37. Satırda klavyeden s değişkenine değer girilmiştir.

2. 38. Satırda bellekte *yeniDugum* için tekrar yer ayrılmıştır.

3. 39. Satırda *yeniDugum->veri=s*; ataması yapılmıştır.

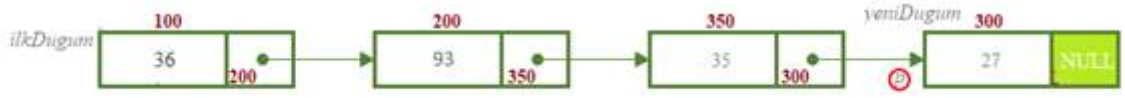
4. 40. Satırda *yeniDugum->sonraki=NULL* ataması yapılmıştır.

5. 41. Satırda *p->sonraki=yeniDugum* atamasıyla döngüye girmeden oluşturulan *p* düğümünün döngü içerisinde oluşan *yeniDugum*' e işaret etmesi sağlanmıştır. Şimdi listenin bellekteki durumu aşağıdaki gibidir.

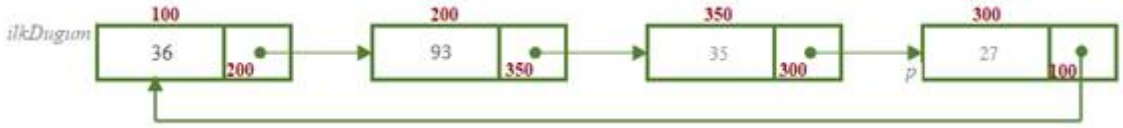


6. *p=p->sonraki* atamasından sonra *yeniDugum* aynı zamanda *p* düğümü olmuştur. Bellekteki son durum aşağıda gösterilmiştir.

7. Programa, oluşturulacak düğüm sayısı olarak 4 (dört) girildiğini varsayarsak, yukarıdaki 6 (altı) adımın 3 (üç) defa tekrarlanacağını görebiliriz. Döngü son defa çalıştıktan sonra, döngüden çıkmadan hemen önce bellekte listenin alacağı durum aşağıda gösterilmiştir.



8. Döngüden çıkıldıktan sonra 41. Satırda *p->sonraki=ilkDugum* ataması yapılarak sondaki *p* düğümünün *ilkDugum* 'ü işaret etmesi sağlanmış ve bellekteki yeni durum aşağıdaki gibi olmuştur.



Oluşan listenin dairesel tek yönlü bağlı liste olduğuna dikkat ediniz.

Not: Programın 4.1 'in açıklanması sırasında kullanılan bellek adresleri semboliktir. Gerçek bellek adreslerini program 4.1 'in ekran çıktısından görebilirsiniz. Sizin bilgisayarlarınızda bu adresler farklı çıkabilir. Bunun nedenini araştırınız.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
 int veri;
 struct dugum *sonraki;
};
struct dugum *ilkDugum=NULL,*yeniDugum,*p;
void dYazdir(int n){
    struct dugum* temp = ilkDugum;int i;
    for(i=0;i<n;i++){
    printf("\n");
    printf("\n*****************************");
    printf("\n");
    printf(" (%d) (%p) (%p)> \n",
    temp->veri,temp->sonraki,&(temp->veri));
    temp = temp->sonraki;
    printf("\n*****************************");
    printf("\n");
    }}
int main() {
    setlocale(LC_ALL,"Turkish");
    int i=0,n,s;
    printf("\nListeye Kaç Dugüm Girilecek : ");
    scanf("%d",&n); printf("\n");
    printf("%d. Düğümün         Verisi   : ",i);
    scanf("%d",&s);
    yeniDugum=(struct dugum*)malloc (sizeof(struct dugum));
    yeniDugum->veri=s;
    yeniDugum->sonraki=NULL;
    ilkDugum=yeniDugum;
    p=yeniDugum;
     for(i=1;i<n;i++){
        printf("%d. Düğümün      Verisi      : ",i);
        scanf("%d",&s);
        yeniDugum=(struct dugum*)malloc(sizeof(struct dugum));
        yeniDugum->veri=s;
        yeniDugum->sonraki=NULL;
        p->sonraki=yeniDugum;
        p=p->sonraki; }
     p->sonraki=ilkDugum;
     dYazdir(n);
     return 0;
     }
```

Program 4.1.

```
Listeye Kaç Dugüm Girilecek : 4

0. Düğümün         Verisi   : 36
1. Düğümün      Verisi      : 93
2. Düğümün      Verisi      : 35
3. Düğümün      Verisi      : 27


*****************************
 (36) (0000000007D7130) (0000000007D7110)>
*****************************


*****************************
 (93) (0000000007D73F0) (0000000007D7130)>
*****************************


*****************************
 (35) (0000000007D7150) (0000000007D73F0)>
*****************************


*****************************
 (27) (0000000007D7110) (0000000007D7150)>
*****************************
```

Program 4.1'in ekran çıktısı

4.1.2. Dairesel Tek Yönlü Bağlı Listenin Başına Eleman Eklemek

Program 4.2 'de, dairesel tek yönlü bağlı listelerin başına yeni eleman eklenmesini sağlayan Program verilmiştir. Program 4.2 yakından incelendiğinde burada tanımlanan yapının, Program 4.1 'de tanımlanan yapı ile aynı olduğu görülecektir.

```
4   struct dugum{
5       int veri;
6       struct dugum *sonraki;
7   };
```

Programda *ilkDugum* ve *SonDugum* değişkenleri, sekizinci satırda global olarak tanımlanmıştır.

```
8   struct dugum *ilkDugum=NULL,*sonDugum=NULL;
```

basaEkle(int s) fonksiyonu tamsayı türünde tek bir parametre almıştır. Fonksiyon ilk defa çağırıldığında eğer bellekte herhangi bir bağlı liste yoksa aşağıdaki ifadeler çalışır ve liste oluşur.

```
10      if(ilkDugum == NULL) {
11          ilkDugum = (struct dugum *)malloc(sizeof(struct dugum));
12          ilkDugum -> veri = s;
13          ilkDugum -> sonraki = ilkDugum;
14      }
```

Liste oluşturulduktan sonra, fonksiyon tekrar çağırıldığında ve çalışma sırası *if* bloğuna geldiğinde, bloğun *else* kısmındaki ifadeler çalışır.

```
15      else {
16          struct dugum *temp = (struct dugum *)malloc(sizeof(struct dugum));
17          struct dugum *sonDugum = ilkDugum;
18          temp -> veri = s;
19          while(sonDugum -> sonraki != ilkDugum){
20              sonDugum = sonDugum -> sonraki;
21          }
```

Bu ifadelerle *önce struct dugum** türden *temp* düğümü oluşturularak *sonDugum* 'e *ilkDugum* 'ün adresi atanır. Daha sonra *temp*'in *veri*'sine parametre değişkeninden gelen *s* aktarılır. En sonunda *while* bloğunda, *sonDugum* döngü içerisinde ilerletilerek listenin son elemanın gösterilmesi sağlanır.

While bloğunun çalışması tamamlandıktan sonra if bloğundan çıkılır ve aşağıdaki ifadeler çalıştırılır.

```
22          temp -> sonraki = ilkDugum;
23          sonDugum -> sonraki = temp;
24          ilkDugum = temp;
```

Yirmi ikinci satırdaki ifade ile *temp, ilkDugum*' ü gösterecek şekilde atama yapılır. Yirmi üçüncü satırda listenin son elemanını

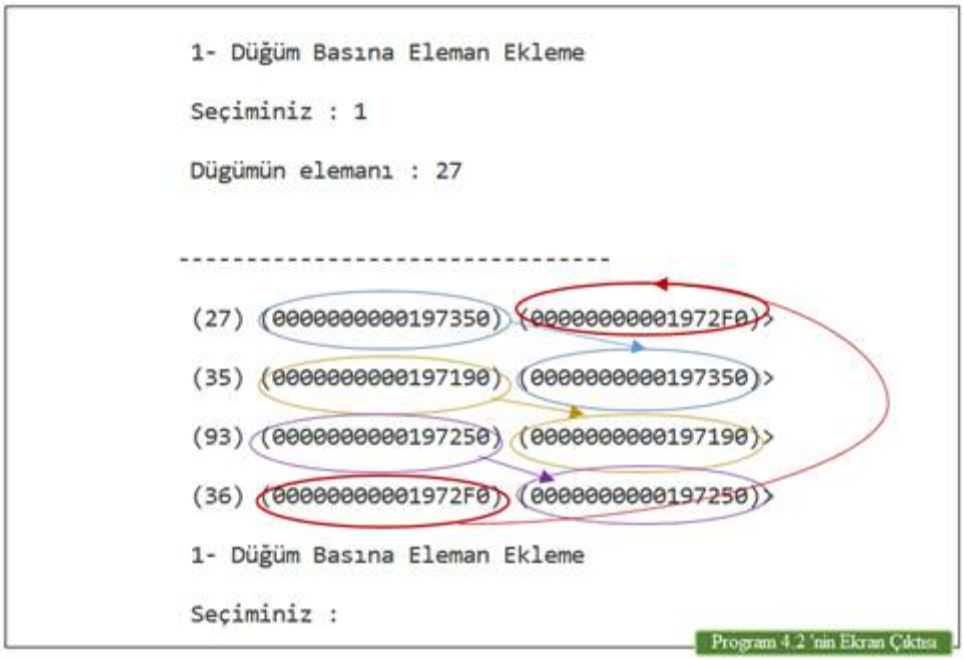
gösteren *sonDugum*'ün *sonraki* işaretçisine *temp*'in adresi atanır ve sonra *ilkDugum*'e *temp* atanarak işlem tamamlanır.

Aşağıda Dairesel tek yönlü bağlı listelerin başına eleman ekleyen program (Program 4.2) verilmiştir.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
};
struct dugum *ilkDugum=NULL,*sonDugum=NULL;
void basaEkle(int s){
    if(ilkDugum == NULL) {
        ilkDugum = (struct dugum *)malloc(sizeof(struct dugum));
        ilkDugum -> veri = s;
        ilkDugum -> sonraki = ilkDugum;
    }
    else {
        struct dugum *temp = (struct dugum *)malloc(sizeof(struct dugum));
        struct dugum *sonDugum = ilkDugum;
        temp -> veri = s;
           while(sonDugum -> sonraki != ilkDugum){
               sonDugum = sonDugum -> sonraki;
           }
    temp -> sonraki = ilkDugum;
    sonDugum -> sonraki = temp;
    ilkDugum = temp;
    }}
void dugumYazdir(){
    int n=0;
    struct dugum *temp = ilkDugum;
    printf("\n            VERİ İŞARETÇİ\n");
    printf("-----------------------\n");
    do{
        n++;
        printf("\n%d. düğüm : (%d)  (%p)\n"
        ,n,temp->veri,temp->sonraki);
        temp = temp->sonraki;
    }while(temp->sonraki!=ilkDugum->sonraki);
}
int main() {
  int secim, s;
  setlocale(LC_ALL,"Turkish");
  while(1){
     printf("\n1-Düğüm Basına Eleman Ekl.");
     printf("\n");
     printf("\n Seçiminiz : ");
     scanf("%d", &secim);
     switch(secim){
     case 1:
      printf("\n Düğümün elemanı : ");
      scanf("%d", &s);
      basaEkle(s);
      dugumYazdir();
      break;
     }
  }
     return 0;
}
```

Program 4.2

Program 4.2 çalıştırıldığında ve klavyeden sırasıyle 36, 93, 35 ve 27 rakamları girildiğinde aşağıdaki çıktıya benzeyen bir ekran çıktısı alabilirsiniz. Çıktının aynısını elde edebilmek için programda *dugumYazdir()* fonksiyonunda küçük değişiklikler yapmalısınız.



Program 4.2 'nin Ekran Çıktısı

4.1.3. Dairesel Tek Yönlü Bağlı Listenin Sonuna Eleman Eklemek

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
};
struct dugum *ilkDugum=NULL,*sonDugum=NULL;
void sonaEkle(int s){
    if(ilkDugum == NULL) {
        ilkDugum = (struct dugum *)
        malloc(sizeof(struct dugum));
        ilkDugum -> veri = s;
        ilkDugum -> sonraki = ilkDugum;
    }
    else {
        struct dugum *temp = (struct dugum *)
        malloc(sizeof(struct dugum));
        struct dugum *sonDugum = ilkDugum;
        temp -> veri = s;

    while(sonDugum-> sonraki != ilkDugum)
        sonDugum = sonDugum -> sonraki;
        temp -> sonraki = ilkDugum;
        sonDugum -> sonraki = temp;
    }
    }
void dugumYazdir(){
    int n=0;
    struct dugum *temp = ilkDugum;
    printf("\n VERİ İŞARETÇİ\n");
    printf("-----------------------\n");
    do{
        n++;
        printf("\n%d. düğüm : (%d)  (%p)\n"
        ,n,temp->veri,temp->sonraki);
        temp = temp->sonraki;
    }while(temp->sonraki!=ilkDugum->sonraki);
}
```

Listede yer alacak düğümlerin yapısı tanımlanıyor. Bu tanımda listede yer alacak düğümlerde tamsayı tipinde bir veri değişkeni ve bir de işaretçi yer alması öngörülmüştür.

ilkDugum ve sonDugum değişkenleri global olarak tanımlanmıştır

Bellekte daha önce liste oluşmamış ise liste oluşturulur. Önce ilkDugum için bellekte yer ayrılır. Sonra klavyeden girilen s değişkeninin değeri ilkDugum->veri 'ye aktarılır.

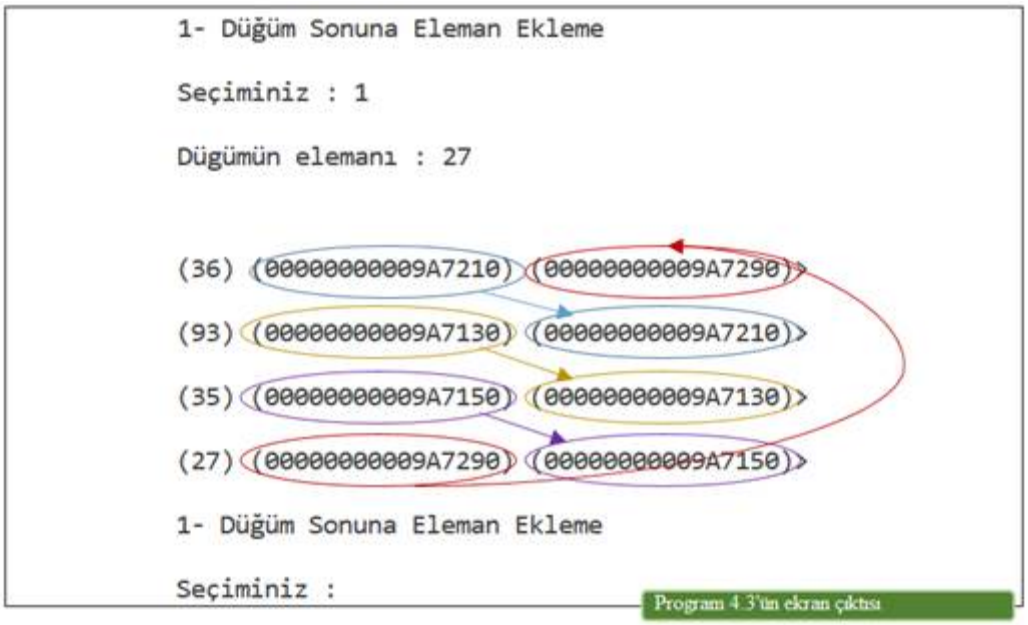
Liste daha önceden oluşmuş ise if bloğunun bu bölümü çalışır. Önce temp adlı değişken tanımlanır ve değşkene bellekte yer ayrılır. Sonra sonDugum tanımlanır sonDugum=ilkDugum, temp->veri=s atamaları yapılır.

while blogu listenin son elemanına bulmak için kullanılır.

temp 'in sonraki işaretçisi ilkDüğümü gösterir
sonDugum 'ün sonraki işaretcisi de temp İ gösterir

```c
int main() {
  int secim, s;
  setlocale(LC_ALL,"Turkish");
  while(1){
   printf("\n1-Düğüm Sonuna Eleman Ekl");
   printf("\n Seçiminiz : ");
   scanf("%d", &secim);
   switch(secim){
    case 1:
     printf("\n Dügümün elemanı:");
     scanf("%d", &s);
     sonaEkle(s);
     dugumYazdir();
     break;
    } }
     return 0;
}
```

Program 4.3



Program 4.3'ün ekran çıktısı

Program 4.3 Dairesel Tek Yönlü Bağlı listelerin sonuna yeni eleman eklemek için C programlama dilinde geliştirilmiştir. Programda yer alan *sonaEkle* fonksiyonu *basaEkle* fonksiyonuna benzemektedir. *basaEkle* fonksiyonunda, fonksiyonun son ifadesi olan *ilkDugum=temp* satırını silerek *sonaEkle* fonksiyonunu elde edebilirsiniz.

4.1.4. Dairesel Tek Yönlü Bağlı Listede Eleman Aramak

Program 4.4'te, *main()* fonksiyonunda öncelikle *sonaEkle* veya *basaEkle* fonksiyonlarından biri çağırılarak dairesel tek yönlü bağlı liste oluşturulmuştur. Liste oluşturulduktan sonra, *main* fonksiyonundaki üçüncü seçenek kullanılarak klavyeden dairesel tek yönlü bağlı listede aranan eleman girilmiş ve tam sayı türündeki bu değişkeni parametre olarak alan *Ara* fonksiyonu çağırılmıştır.

Ara fonksiyonunda 51. Satırda *temp* değişkeni oluşturulmuş ve ilkDugum temp 'e atanmıştır.

```
51 |    struct dugum *temp = ilkDugum;
```

Sonra 54-62. Satırlar arasında do-while döngüsü oluşturulmuş ve döngü içerisinde ilk düğümden başlayıp son düğüme kadar bütün elemanları dolaşacak bir yapı, *traversal (dolaşma)* yapısı yazılmıştır.

```
54      do{
55          n++;
56          if(temp->veri1==s ||temp->veri2==s){
57              printf("\n Aranan sayı %d. düğümde bulundu : (%d  %d)  (%p)\n",
58              n,temp->veri1,temp->veri2,temp->sonraki);
59              return;
60              }
61          temp = temp->sonraki;
62      }while(temp->sonraki != ilkDugum->sonraki);
```

Döngü içerisinde 56-60. Satırlar arasında aranan elemanı bulup ekrana yazdırmak için if bloğu oluşturulmuştur.

```
56          if(temp->veri1==s ||temp->veri2==s){
57              printf("\n Aranan sayı %d. düğümde bulundu : (%d  %d)  (%p)\n",
58              n,temp->veri1,temp->veri2,temp->sonraki);
59              return;
60              }
```

main fonksiyonunda, klavyeden aranan eleman olarak girilen ve *Ara* fonksiyonuna parametre olarak atanan s değişkeni, düğümlerde yer alan *veri1* ve *veri2* değerleri ile karşılaştırılmış ve s' e eşit bir değer bulunduğunda kullanıcıya aranan elemanın listenin kaçıncı elemanı olduğu bilgisi verilerek *return* ile döngüden çıkılması sağlanmıştır.

SIRA SİZDE 1

Program 4.4' te yazılan Ara fonksiyonunu geliştirerek dairesel tek yönlü bağlı listenin elemanlarında, aranan değerin birden fazla olması durumunda, aranan değerin bulunduğu elemanlarla ilgili bilginin tamamını ekrana yazdırılmasını sağlayınız

SIRA SİZDE 2

Program 4.1 de, aranan sayının liste elemanları arasında bulunmaması durumunda kullanıcıya ekrandan uygun mesajın verilmesini sağlayacak geliştirmeyi yapınız

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri1;
    int veri2;
    struct dugum *sonraki;
};
struct dugum *ilkDugum=NULL,*sonDugum=NULL;
void sonaEkle(int s1,int s2){//Bkz. Program 4.3
void basaEkle(int s1,int s2){//Bkz. Program 4.2
void Ara(int s){
 int n=0;
 struct dugum *temp = ilkDugum;
 printf("\n           VERİ İŞARETÇİ\n");
 printf("---------------------------\n");
 do{
  n++;
  if(temp->veri1==s ||temp->veri2==s){
  printf("\n Aranan sayı %d.düğümde bulundu:(%d  %d) (%p)\n",
                n,temp->veri1,temp->veri2,temp->sonraki);
                return;
                }
        temp = temp->sonraki;
    }while(temp->sonraki != ilkDugum->sonraki);
}
void dugumYazdir(){//Bkz.Program 4.2
```

```c
int main() {
int secim, s1,s2,s;
setlocale(LC_ALL,"Turkish");
while(1){
    printf("\n 1- Düğüm Sonuna Eleman Ekleme");
    printf("\n 2- Düğüm Başına Eleman Ekleme");
    printf("\n 3- Düğümde Eleman Arama");
    printf("\n Seçiminiz : ");
    scanf("%d", &secim);
    switch(secim){
    case 1:
        printf("\n Düğümün  birinci elemanı : ");
        scanf("%d", &s1);
        printf("\n Düğümün  ikinci  elemanı : ");
        scanf("%d", &s2);
        sonaEkle(s1,s2);
        dugumYazdir();
        break;
    case 2:
        printf("\n Düğümün  birinci elemanı : ");
        scanf("%d", &s1);
        printf("\n Düğümün  ikinci  elemanı : ");
        scanf("%d", &s2);
        basaEkle(s1,s2);
        dugumYazdir();
        break;
    case 3:
        printf("\n Arana Eleman : ");
        scanf("%d", &s);
        Ara(s);
        break; } }
return 0;}
```

Program 4.4

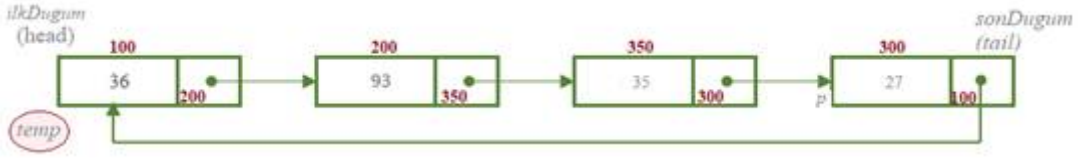
```
1. düğüm : (11  22)  (0000000000027250)

2. düğüm : (33  44)  (00000000000272D0)

3. düğüm : (55  66)  (0000000000027270)

4. düğüm : (77  88)  (00000000000272B0)

 1- Düğüm Sonuna Eleman Ekleme

 2- Düğüm Başına Eleman Ekleme

 3- Düğümde Eleman Arama

 Seçiminiz : 3

 Arana Eleman : 55

           VERİ İŞARETÇİ
--------------------------------

 Aranan sayı 3. düğümde bulundu :
 (55  66)  (0000000000027270)
```

Program 4.4 in ekran çıktı

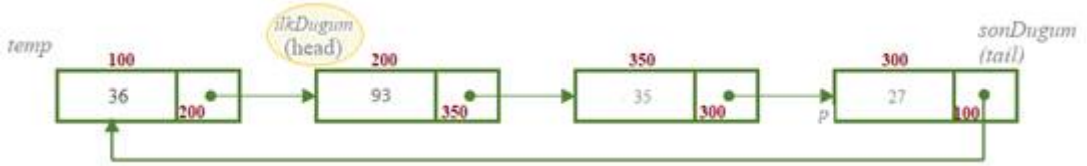
4.1.5. Dairesel Tek Yönlü Bağlı Listenin Başından Eleman Silmek

Listenin başındaki elemanı silmek için sırasıyla aşağıdaki işlemler yapılır. Aşağıda her işlemin sonunda listenin yeni durumu şekiller üzerinde gösterilmiştir.

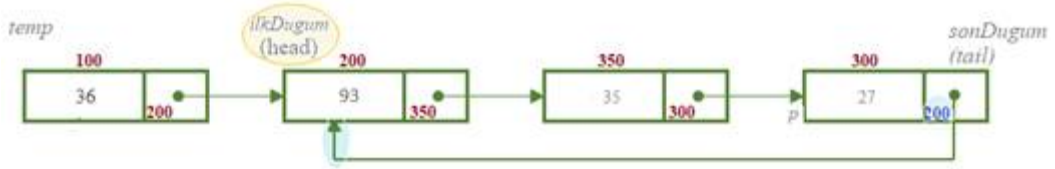
```
1.  temp = ilkDugum;
```



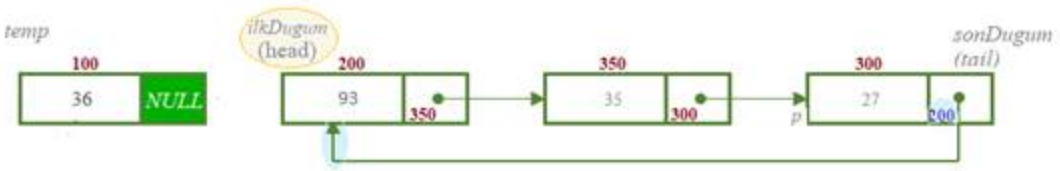

```
2.  ilkDugum=ilkDugum->sonraki;
```



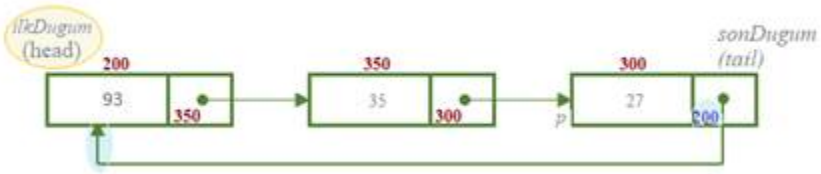

```
3.  sonDugum->sonraki=ilkDugum;
```




```
4.  temp->sonraki=NULL;
```




```
5.  free(temp);
```



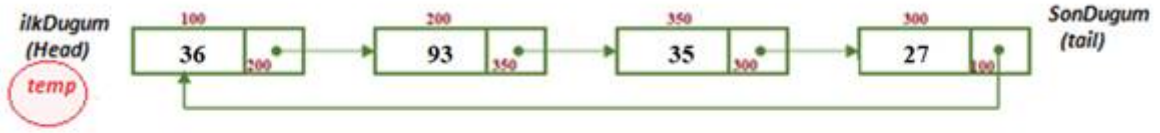
```
 9  void bastanSil( ) {
10      struct dugum *temp;
11      temp=ilkDugum;
12      while(sonDugum -> sonraki != ilkDugum)
13          sonDugum = sonDugum-> sonraki;
14      ilkDugum = ilkDugum -> sonraki;
15      sonDugum->sonraki=ilkDugum;
16      temp->sonraki=NULL;
17      free(temp);}
```

bastanSil() fonksiyonu

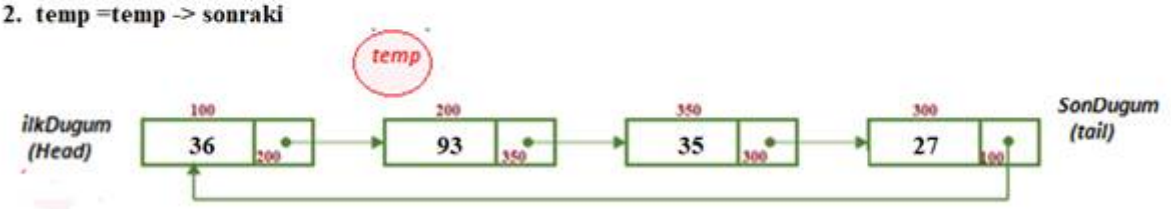
4.1.6. Dairesel Tek Yönlü Bağlı Listenin Arasından Eleman Silmek

Dairesel tek yönlü bağlı listeden indeks değeri (listedeki sıra numarası) iki olan üçüncü sıradaki düğümü silmek istediğimizi varsayarsak, aşağıdaki işlem adımlarını takip edebiliriz. İşlem adımlarını ve bu işlemler için yazılmış olan fonksiyonu inceleyiniz.

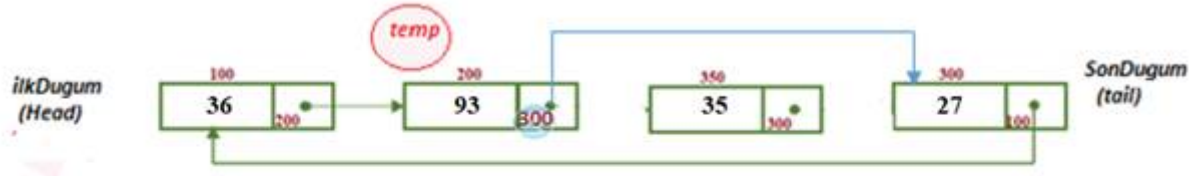
1. temp = ilkDugum



2. temp =temp -> sonraki



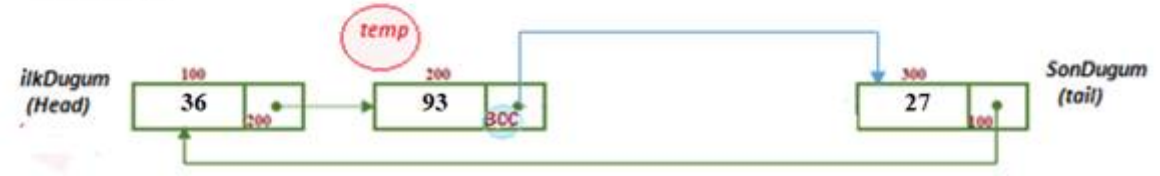
3. temp -> sonraki = temp -> sonraki -> sonraki



4. temp2 = temp -> sonraki

. . . . . .

free(temp2)



```
void aradanSil(int pos ) {
    if(pos==0)
    {
        printf("\nilk elemanı silmek için menüden 2. seçeneği kullanınız\n");
        return;
    }
    struct dugum *temp,*temp2;int i;
    temp=ilkDugum;
    for(i=0;i<pos-1;i++){
        temp=temp->sonraki;
    }
    temp2=temp->sonraki;
    temp->sonraki=temp->sonraki->sonraki;
    free(temp2);
        }
```

4.2. Dairesel Çift Yönlü Bağlı Listeler

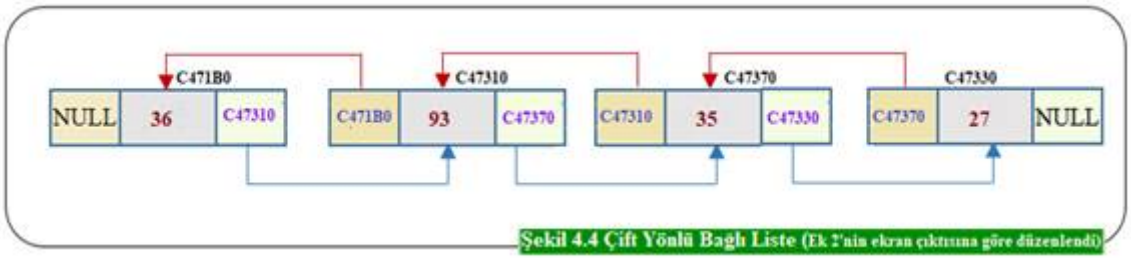
Üçüncü bölümde çift yönlü bağlı listelerde, listenin ilk düğümünün önceki *(prev)* işaretçisinin değeri ve de son düğümünün sonraki (*next*) işaretçisinin değerinin *NULL* olduğunu öğrenmiştik. Dairesel çift yönlü bağlı listelerde ise listenin ilk düğümünün önceki (*prev*) işaretçisinin değeri, sonuncu düğümün veri (*data*) kısmını ve son düğümünün sonraki (*next*) işaretçisinin değeri listenin ilk düğümünün veri (data) kısmını göstermektedir. ***Bkz. Şekil 4.3.***

Bolum sonunda **EK2**'de verile C programı **Çift yönlü bağlı liste** oluşturan bir programdır. Programın ekran çıktısı aşağıdaki gibidir:

```
Düğüm Sayısı   :4
Düğümün Verisi :36
Düğümün Verisi :93
Düğümün Verisi :35
Düğümün Verisi :27


 ÖNCEKİ             VERİ      SONRAKİ          VERİNİN ADRESİ
(0000000000000000) (36) (0000000000C47310) (0000000000C471B0)
(0000000000C471B0) (93) (0000000000C47370) (0000000000C47310)
(0000000000C47310) (35) (0000000000C47330) (0000000000C47370)
(0000000000C47370) (27) (0000000000000000) (0000000000C47330)
```

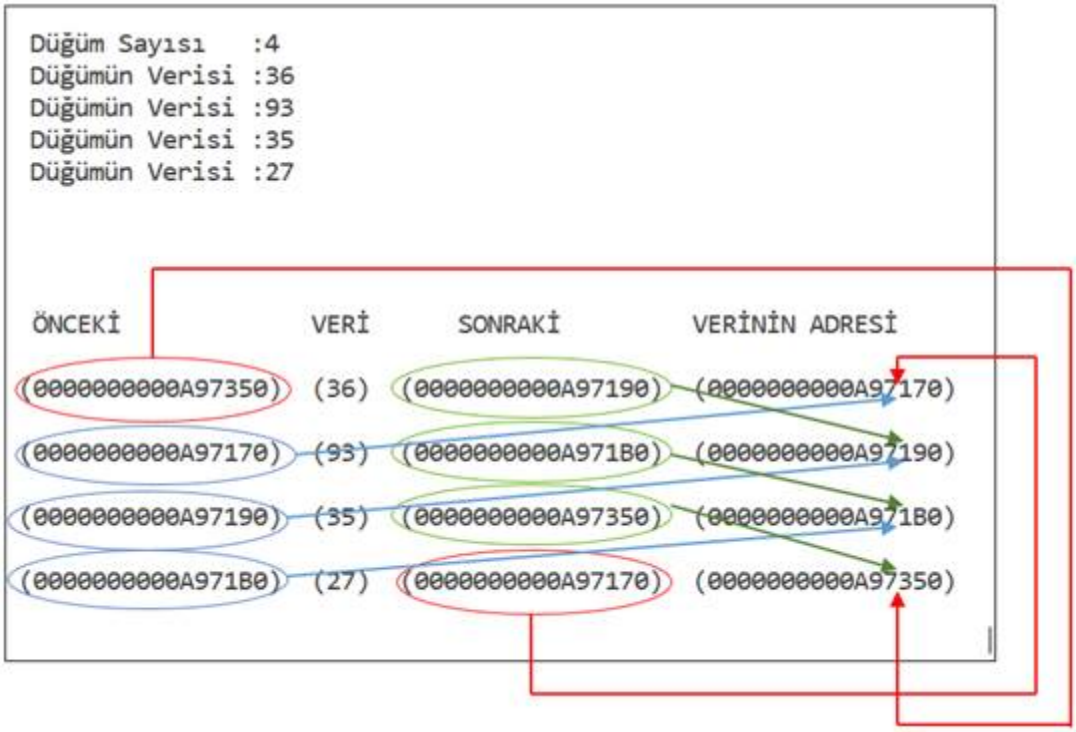
Hatırlanacağı gibi çift yönlü bağlı listelerin iki temel özelliği vardır. Bu özelliklerden birincisi listedeki her düğüm ilki *sonraki (next)*, diğeri *onceki (prev)* olmak üzere iki işaretçiye sahiptir. İkinci temel özellik sonDugum 'ün sonraki (next) işaretçisi ve ilkDugum' ün onceki (prev)

işaretçisi NULL değeri alır. Bu durum ekran çıktısında kırmızı renkte daire içerisine alınarak gösterilmiştir. Yukarıda verilen ekran çıktısı şekil 4.4 'te gösterilmiştir.



Şekil 4.4 Çift Yönlü Bağlı Liste (Ek 2'nin ekran çıktısına göre düzenlendi)

Bolum sonunda **EK3**'de verile C programı **Dairesel Çift yönlü bağlı liste** oluşturan bir programdır. Programın ekran çıktısı aşağıdaki gibidir:



Yukarıda verilen ekran çıktısı incelendiğinde;

1. Dairesel çift yönlü bağlı listelerde *sonDugum (tail)* 'ün *sonraki (next)* işaretçisinin *ilkDugum (head)* 'ün adresini gösterdiğini,

2. Dairesel çift yönlü bağlı listelerde *ilkDugum (head)* 'ün *onceki (prev)* işaretçisinin *sonDugum (tail)* 'ün adresini gösterdiğini söyleyebiliriz.

EK3 'te verilen ve dairesel çift yönlü liste oluşturmak için yazılan C programının, yukarıda sunulan ekran çıktısı değerleri ile düzenlenmiş hali şekil 4.5 'te verilmiştir.

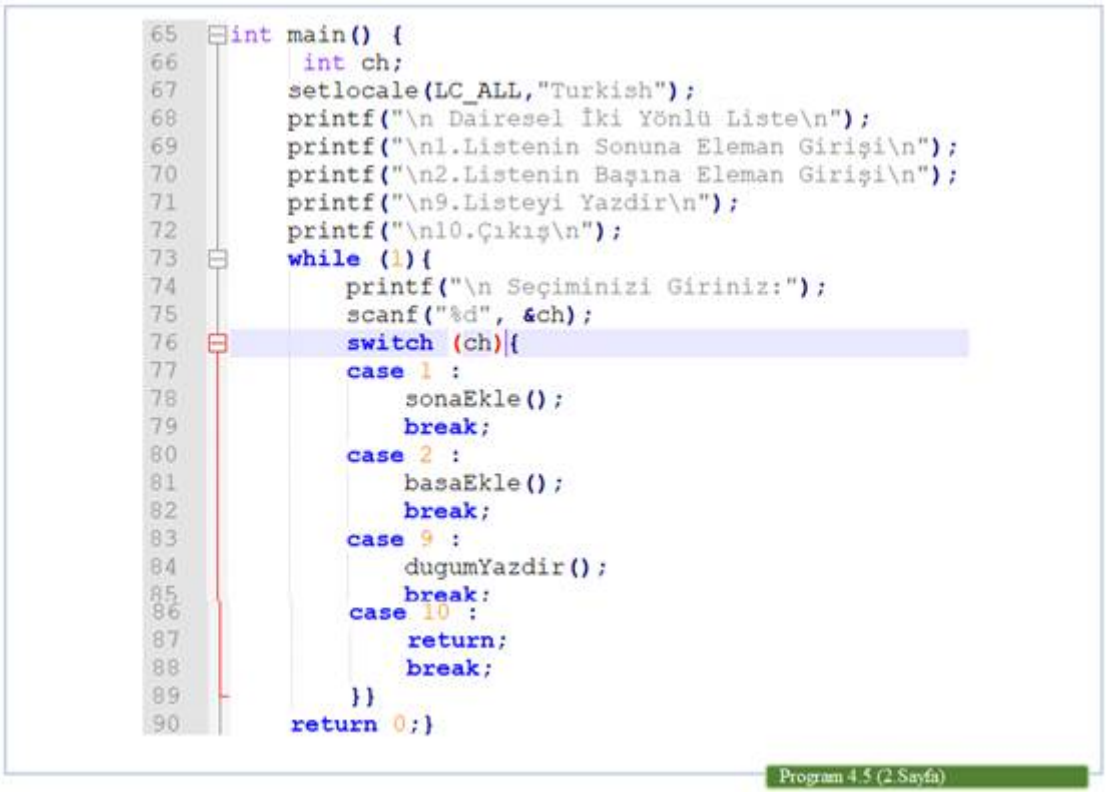
Şekil 4.5 Dairesel Çift Yönlü Bağlı Liste (Yukarıdaki ekran çıktısı verileri ile diz.)

4.2.1. Dairesel Çift Yönlü Bağlı Listelerin Sonuna Eleman Eklemek

Dairesel çift yönlü bağlı listenin sonuna bir düğüm ekleyen C programı Program 4.5 'te verilmiştir.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
    struct dugum *onceki;
};
struct dugum *ilkDugum=NULL,*sonDugum=NULL,*yeniDugum,*ptr;
int sayi=0;
struct dugum *olustur(int s){
    sayi++;
    yeniDugum = (struct dugum*)malloc(sizeof(struct dugum));
    yeniDugum->veri = s;
    yeniDugum->sonraki = NULL;
    yeniDugum->onceki = NULL;
    return yeniDugum; }
void sonaEkle(){
     int s;
    printf("\nDüğümün Verisini Giriniz :");
    scanf("%d", &s);
    yeniDugum = olustur(s);
     if (ilkDugum == sonDugum && ilkDugum == NULL){
        ilkDugum = sonDugum = yeniDugum;
        ilkDugum->sonraki = sonDugum->sonraki = NULL;
        ilkDugum->onceki = sonDugum->onceki = NULL;
    }
    else{
        sonDugum->sonraki = yeniDugum;
        yeniDugum->onceki = sonDugum;
        sonDugum = yeniDugum;
        ilkDugum->onceki = sonDugum;
        sonDugum->sonraki = ilkDugum;
    }}

void dugumYazdir(){
    int i;
    if (ilkDugum == sonDugum && ilkDugum == NULL)
        printf("\nListe Bos");
    else{
        printf("\n Listede Bulunan Dugum Sayısı :%d\n", sayi);
        printf
        ("\n  ÖNCEKİ             VERİ      SONRAKİ         VERİNİN ADRESİ\n");
        for (ptr = ilkDugum, i = 0;i < sayi;i++,ptr = ptr->sonraki)
            printf("\n (%p)   (%d)   (%p)   (%p)    ",
            ptr->onceki, ptr->veri,ptr->sonraki,&(ptr->veri));
        printf("\n");
    }}
```

Program 4.5 (1. Sayfa)



```
int main() {
    int ch;
    setlocale(LC_ALL,"Turkish");
    printf("\n Dairesel İki Yönlü Liste\n");
    printf("\n1.Listenin Sonuna Eleman Girişi\n");
    printf("\n2.Listenin Başına Eleman Girişi\n");
    printf("\n9.Listeyi Yazdir\n");
    printf("\n10.Çıkış\n");
    while (1){
        printf("\n Seçiminizi Giriniz:");
        scanf("%d", &ch);
        switch (ch){
        case 1 :
            sonaEkle();
            break;
        case 2 :
            basaEkle();
            break;
        case 9 :
            dugumYazdir();
            break;
        case 10 :
            return;
            break;
        }}
    return 0;}
```

Program 4.5 (2.Sayfa)

4.2.2. Dairesel Çift Yönlü Bağlı Listelerin Başına Eleman Eklemek

Dairesel çift yönlü bağlı listenin başına bir düğüm ekleyen C programı Program 4.6 'da verilmiştir.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
    struct dugum *onceki;};
struct dugum *ilkDugum=NULL,*sonDugum=NULL,*yeniDugum,*ptr;
int sayi=0;
struct dugum *olustur(int s){
    sayi++;
    yeniDugum = (struct dugum*)malloc(sizeof(struct dugum));
    yeniDugum->veri = s;
    yeniDugum->sonraki = NULL;
    yeniDugum->onceki = NULL;
    return yeniDugum;}

void basaEkle(){
    int s;
    printf("\nDüğümün Verisini Giriniz :");
    scanf("%d", &s);
    yeniDugum = olustur(s);
    if (ilkDugum == sonDugum && ilkDugum == NULL){
        ilkDugum = sonDugum = yeniDugum;
        ilkDugum->sonraki = sonDugum->sonraki = NULL;
        ilkDugum->onceki = sonDugum->onceki = NULL;}
    else{
        yeniDugum->sonraki = ilkDugum;
        ilkDugum->onceki = yeniDugum;
        ilkDugum = yeniDugum;
        ilkDugum->onceki= sonDugum;
        sonDugum->sonraki = ilkDugum;}}
```

Program 4.6 (sayfa 1)

```c
void dugumYazdir(){
    int i;
    if (ilkDugum == sonDugum && ilkDugum == NULL)
        printf("\nListe Bos");
    else{
     printf
    ("\n  ÖNCEKİ              VERİ       SONRAKİ          VERİNİN ADRESİ\n");
        for (ptr = ilkDugum, i = 0;i < sayi;i++,ptr = ptr->sonraki)
            printf("\n (%p)   (%d)   (%p)   (%p)    "
           ,ptr->onceki, ptr->veri,ptr->sonraki,&(ptr->veri));
        printf("\n"); }}
int main() {
     int ch;
    setlocale(LC_ALL,"Turkish");
    printf("\n Dairesel İki Yönlü Liste\n");
    printf("\n1.Listenin Başına Eleman Girişi\n");
    printf("\n9.Listeyi Yazdir\n");
    printf("\n10.Çıkış\n");
    while (1){
        printf("\n Seçiminizi Giriniz:");
        scanf("%d", &ch);
        switch (ch){
        case 1 :
            basaEkle();
            break;
        case 9 :
            dugumYazdir();
            break;
        case 10 :
            return;
            break; }}
    return 0;}
```

Program 4.6 (sayfa 2)

SIRA SİZDE 3 — Dairesel çift yönlü listelerde klavyeden girilecek bir değikene ait değerin listenin kaçıncı elemanında yer aldığını ekrana yazdıran C programı yazınız?

SIRA SİZDE 4 — Dairesel çift yönlü bağlı listelerde klavyeden girilecek sıradaki elemanın silinmesini sağlayacak C programını yazınız?

EKLER

EK1:

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
 int veri;
 struct dugum *sonraki;};
struct dugum *ilkDugum=NULL,*yeniDugum,*p;
void dYazdir(int n){
    struct dugum* temp = ilkDugum;int i;
    for(i=0;i<n;i++){
    printf("\n");
    printf("\n*****************************");
    printf("\n");
    printf(" (%d) (%p) (%p)> \n",
    temp->veri,temp->sonraki,&(temp->veri));
    temp = temp->sonraki;
    printf("\n*****************************");
    printf("\n");}}
int main() {
    setlocale(LC_ALL,"Turkish");    int i=0,n,s;
    printf("\nListeye Kaç Dugüm Girilecek : ");
    scanf("%d",&n);
    printf("\n");
    printf("%d. Düğümün          Verisi    : ",i);
    scanf("%d",&s);
    yeniDugum=(struct dugum*)malloc (sizeof(struct dugum));
    yeniDugum->veri=s;
    yeniDugum->sonraki=NULL;
    ilkDugum=yeniDugum;
    p=yeniDugum;
    for(i=1;i<n;i++){
       printf("%d. Düğümün       Verisi       : ",i);
       scanf("%d",&s);
       yeniDugum=(struct dugum*)malloc(sizeof(struct dugum));
       yeniDugum->veri=s;
       yeniDugum->sonraki=NULL;
       p->sonraki=yeniDugum;
       p=p->sonraki;}
       dYazdir(n);
    return 0;}
```

EK 1: Tek yönlü bağlı liste oluşturmak için yazılan program

EK2:

EK 2 : Çift yönlü bağlı liste oluşturumak için yazılan C programı

(Sayfa 1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
    struct dugum *onceki;
};
struct dugum *ilkDugum=NULL,*sonDugum=NULL,*ptr;
int sayi=0;
void dugumYazdir(int n){
    int i;
    if (ilkDugum == sonDugum && ilkDugum == NULL)
        printf("\nListe Bos");
    else{
        printf("\n Listede Bulunan Dugum Sayısı :%d\n", sayi);
        printf
        ("\n  ÖNCEKİ             VERİ      SONRAKİ         VERİNİN ADRESİ\n");
        for (ptr = ilkDugum, i = 0;i < n;i++,ptr = ptr->sonraki)
            printf("\n (%p)   (%d)   (%p)   (%p)    \n",
            ptr->onceki, ptr->veri,ptr->sonraki,&(ptr->veri));
        printf("\n");
    }}



    int main() {
        setlocale(LC_ALL,"Turkish");
        int n,i,s;
        struct dugum *yeniDugum,*p;
        printf(" Düğüm Sayısı    :");
        scanf("%d",&n);
        printf(" Düğümün Verisi :");
        scanf("%d",&s);
        yeniDugum=(struct dugum *)malloc(sizeof(struct dugum));
        yeniDugum->veri=s;
        yeniDugum->sonraki=NULL;
        yeniDugum->onceki=NULL;
        ilkDugum=yeniDugum;
        p=yeniDugum;
        for(i=1;i<n;i++)
        {
        printf(" Düğümün Verisi :");
        scanf("%d",&s);
        yeniDugum=(struct dugum *)malloc(sizeof(struct dugum));
        yeniDugum->veri=s;
        yeniDugum->sonraki=NULL;
        yeniDugum->onceki=p;
        p->sonraki=yeniDugum;
        p=p->sonraki;
        }
        dugumYazdir(n);
        return 0;
    }
```

EK: 3

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct dugum{
    int veri;
    struct dugum *sonraki;
    struct dugum *onceki;};
struct dugum *ilkDugum=NULL,*sonDugum=NULL,*ptr;
int sayi=0;
void dugumYazdir(int n){
    int i;
    if (ilkDugum == sonDugum && ilkDugum == NULL)
        printf("\nListe Bos");
    else{
        printf
        ("\n  ÖNCEKİ              VERİ      SONRAKİ          VERİNİN ADRESİ\n");
        for (ptr = ilkDugum, i = 0;i < n;i++,ptr = ptr->sonraki)
            printf("\n (%p)   (%d)   (%p)   (%p)    \n",
            ptr->onceki, ptr->veri,ptr->sonraki,&(ptr->veri));
            printf("\n");}}
    int main() {
        setlocale(LC_ALL,"Turkish");
        int n,i,s;
        struct dugum *yeniDugum,*p;
        printf(" Düğüm Sayısı   :");
        scanf("%d",&n);
        printf(" Düğümün Verisi :");
        scanf("%d",&s);
        yeniDugum=(struct dugum *)malloc(sizeof(struct dugum));
        yeniDugum->veri=s;
        yeniDugum->sonraki=NULL;
        yeniDugum->onceki=NULL;
        ilkDugum=yeniDugum;
        p=yeniDugum;
        for(i=1;i<n;i++){
        printf(" Düğümün Verisi :");
        scanf("%d",&s);
        yeniDugum=(struct dugum *)malloc(sizeof(struct dugum));
        yeniDugum->veri=s;
        yeniDugum->sonraki=NULL;
        yeniDugum->onceki=p;
        p->sonraki=yeniDugum;
        p=p->sonraki;}
        p->sonraki=ilkDugum;
        ilkDugum->onceki=p;
        dugumYazdir(n);
        return 0; }
```

EK 3: Dairesel çift yönlü bağlı liste oluşturmak için yazılan C programı

Bölüm Özeti

Dairesel Tek Yönlü Bağlı listelerin yapısını açıklayabilecek,

Dairesel tek yönlü bağlı listelerin yapısı tanımlanırken, tanımda doğrusal olan tek yönlü bağlı listelerin tanımından farklı herhangi bir işlem yapılmamaktadır. Her iki tanımlamada da aşağıdaki yapı kullanılmaktadır.

```
struct dugum {
    int veri;                   // Data
    struct dugum *sonraki;      // Adres
};
```

Dairesel Tek Yönlü Bağlı Listelerin Veri Yapısı

Fakat dairesel tek yönlü bağlı listelerle, doğrusal olan tek yönlü bağlı listeler arasında önemli bir fark vardır. Bu fark dairesel tek yönlü bağlı listelerin sonDugum' ünde bulunan işaretçisinin değerinin NULL olmayıp listenin ilkDugum (head) 'üne işaret etmesidir. Dolayısı ile buradan hareketle, dairesel tek yönlü bağlı listelerde son düğümün bulunmadığı ve bu listelerin sonlanmadığı söyleyebilir.

Dairesel Tek Yönlü Bağlı liste oluşturup listeye eleman eklenmesini sağlayan C programı yazabilecek,

Dairesel tek yönlü bağlı liste oluşturacak, bu listeye eleman ekleyecek ve listenin elemanlarını yazdıracak C programı yazarken yukarıda belirtilen farklılık hep göz önünde bulundurulmalıdır. Aksi halde beklenen çıktıyı sağlayan bir program yazılamaz. Fakat her iki listeyi oluştururken yazılacak programlar arasındaki bu farklılığın çok büyük olmadığını Program 4.1 ve Ek 1 de verilen C programları incelenerek görülebilir.

Dairesel Tek Yönlü Bağlı Listelerde eleman aranmasını, listeden elaman silinmesini saylayan C programları yazabilecek,

C dilinde, klavyeden girilen bir değişkene ait değerin, dairesel bağlı listelerde kaçıncı elemanda (düğümde) olduğunu bulmak için program 4.4 'te verilen örnekte arama işlemi do – while döngüsü içerisinde yapılmıştır. Listede aranan değer bulunduğunda sonuç ekrana yazılmış ve programın çalışması sonlandırılmıştır. Bu işlem için başka çözümler de üretilebileceği göz önünde bulundurulmalıdır.

Bu bölümde, dairesel tek yönlü bağlı listelerin başından ve listede arada bulunan bir elemanı silmeye örnek olacak iki C programı yazılmış ve bu programların çalışma şekli de detaylı olarak açıklanmıştır.

Dairesel İki Yönlü Bağlı listelerin yapısını açıklayabilecek

Dairesel iki yönlü bağlı liste tanımı iki yönlü bağlı liste tanımına benzer şekilde yapılmaktadır.

```
struct dugum{
    int veri;
    struct dugum *sonraki;
    struct dugum *onceki;
};
```

Aralarındaki fark, çift yönlü bağlı listelerse ilkDugum 'ün onceki(prev) işaretçisi NULL, sonDugum 'ün sonraki (next) işaretçisi de NULL 'dır. Dairesel çift yönlü bağlı listelerde ise *sonDugum* 'ün *sonraki (next)* işaretçisi *ilkDugum (head)* 'ün adresini göstermekte ve *ilkDugum (head)* 'ün *onceki (prev)* işaretçisi *sonDugum (tail)* 'ün adresini göstermektedir.

Dairesel İki Yönlü Bağlı liste oluşturup listeye eleman eklenmesini sağlayan C programı yazabilecek,

Bu bölümün son konusu olarak dairesel çift yönlü listelere eleman eklenmesi ele alınmış ve konu bu başlık altında üç ayrı C programına yer verilerek açıklanmıştır.