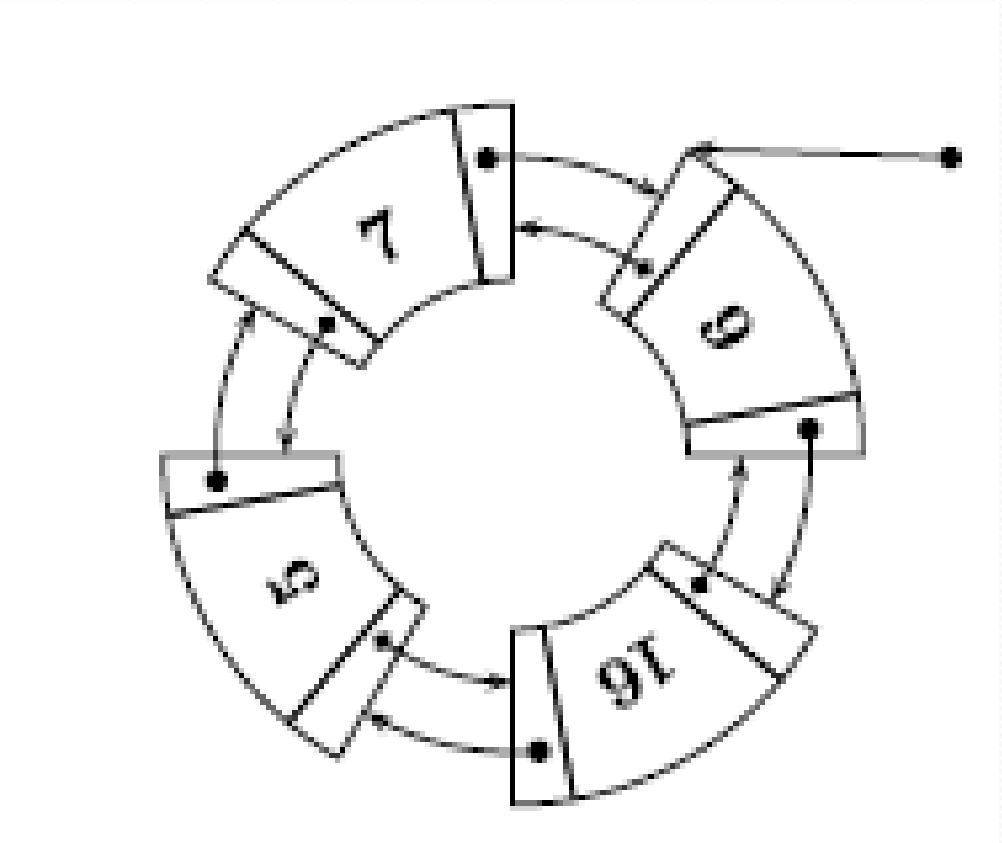
CME 112- Programming Languages II
Lecture 5: Linked Lists (Part-2)

Assist.Prof.Dr. Ümit ATİLA

CIRCULAR LINKED LISTS

- Nodes are linked in one direction in the list
- Difference from singly linked list is that last node shows the address of head node in the list.
- We can access all nodes in the list through the address of any node in the list.

CIRCULAR LINKED LISTS



Head of List

Node Structure

```
 4  struct ogrenci{
 5      int no;
 6      char adi[40];
 7      int yas;
 8      struct ogrenci *sonraki;
 9  };
10  typedef struct ogrenci dugum;
11  dugum *head,*yeni;
```

Create List

```
13  void listeOlustur()
14  {
15      int n,k;
16      printf("Kac elamanli liste olusturacaksin");
17      scanf("%d",&n);
18      for(k=0;k<n;k++)
19      {
20          if(k==0) //ilk düğüm ekleniyor
21          {
22              yeni = (dugum *) malloc(sizeof(dugum));
23              head = yeni;
24          }
25          else
26          {
27              yeni->sonraki = (dugum *) malloc(sizeof(dugum));
28              yeni = yeni->sonraki;
29          }
30          scanf("%d %s %d",&yeni->no,yeni->adi,&yeni->yas);
31      }
32      yeni->sonraki = head;
33  }
```

Traversing List

```
35  void listeDolas()
36  {
37      int sayac =1;
38      dugum *p;
39      p = head;
40      while(1)
41      {
42          printf("%d- %d %s %d \n",sayac,p->no,p->adi,p->yas);
43          sayac++;
44          if(p->sonraki==head)
45              break;
46          else
47              p = p->sonraki;
48      }
49  }
```

Insert Node

```
51  void dugumEkle()
52  {
53      int rno;
54      dugum *p,*q,*yeniDugum;
55      yeniDugum = (dugum *) malloc(sizeof(dugum));
56      printf("Yeni dugum icin veri gir");
57      scanf("%d %s %d",&yeniDugum->no,yeniDugum->adi,&yeniDugum->yas);
58
59      printf("Hangi kayittan oncesine eklemek istiyorsunuz");
60      printf("Liste sonuna eklemek icin 0 gir");
61      scanf("%d",&rno);
62
63      p = head;
64      if (p->no == rno) /* Başa ekleme */{
65          q=p;
66          p=p->sonraki;
67          while(p->sonraki!=q)
68              p=p->sonraki;
69          if(p->sonraki==q){
70              yeniDugum->sonraki = q;
71              p->sonraki = yeniDugum;
72              head = yeniDugum;
73          }
74      }
```

Insert Node

```
75      else {
76          while (p->no != rno){
77              q = p;
78              p = p->sonraki;
79              if(p == head)
80                  break;
81          }
82          if (p == head)/* Sona ekleme */{
83              q->sonraki = yeniDugum;
84              yeniDugum->sonraki = head;
85          }
86          else if (p->no == rno)/* Araya ekleme */{
87              q->sonraki = yeniDugum;
88              yeniDugum->sonraki = p;
89          }
90      }
91  }
```

Delete Node

```
 93 void dugumSil()
 94 {
 95     int rno;
 96     dugum *p, *q;
 97     printf("\nDelete for no :");
 98     scanf("%d", &rno);
 99     p = head;
100     if (p->no == rno)/* ilk düğümü sil */{
101         q=p;
102         p=p->sonraki;
103         while(p->sonraki!=q)
104             p=p->sonraki;
105         if(p->sonraki==q){
106             p ->sonraki = q->sonraki;
107             head=q->sonraki;
108             free(q);
109         }
110     }
```

Delete Node

```
111     else {
112         while (p->no != rno){
113             q = p;
114             p = p->sonraki;
115             if(p == head)
116                 break;
117         }
118         if (p == head)/* Silinecek düğüm bulunamadı */
119             printf("\nSilinecek düðüm bulunamadý");
120         else if (p->no == rno){
121             q->sonraki = p->sonraki;
122             free (p);
123         }
124     }
125 }
```

Circular Linked List Application

```
127 int main(void)
128 {
129     int secim=0;
130     head = NULL, yeni = NULL;
131     printf("Tek bağlı dairesel\n");
132     printf("1-Liste Olustur 2-Liste Dolas 3-Dugum Sil 4-Dugum Ekle 5-Cikis\n");
133     while(1)
134     {
135         printf("Secim yap [1-5]?");
136         scanf("%d",&secim);
137         switch(secim)
138         {
139         case 1: listeOlustur();
140             listeDolas();break;
141         case 2: listeDolas();break;
142         case 3: dugumSil();
143             listeDolas();break;
144         case 4: dugumEkle();
145             listeDolas();break;
146         case 5: exit(0);
147         }
148     }
149 }
```

Doubly Linked Lists

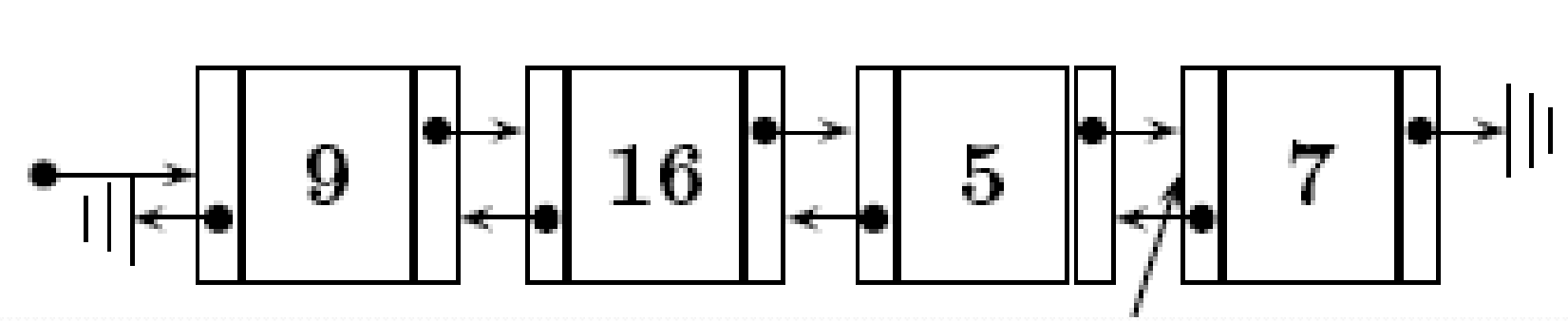
- Although inserting or deleting elements in linked lists are easier than arrays they also have some disadvantages. :
  - We only can move forward in singly linked lists.

Doubly Linked Lists

- This disadvantage effects the operations given below:
  - We only can insert an element into the list or delete an element from the list if only we know the previous or next element of these elements.
  - If we dont have the pointer that shows the previous element, we have to traverse through the list from head node to the location that we want to insert node or delete node.

Doubly Linked Lists

- This disadvantage can be overcame with a pointer that shows the previous node for each node in the list.
- The list having nodes with two pointers is called Doubly linked List.



Structure of Node with Two Pointers

```
 4 struct ogrenci{
 5       int no;
 6       char adi[40];
 7       int yas;
 8       struct ogrenci *sonraki;
 9       struct ogrenci *onceki;
10  };
11 typedef struct ogrenci dugum;
12  dugum *head,*tail,*yeni;
```

Creating List

```
14  void listeOlustur()
15  {
16      int n,k;
17      printf("Kac elamanli liste olusturacaksin");
18      scanf("%d",&n);
19      for(k=0;k<n;k++)
20      {
21          if(k==0) //ilk düğüm ekleniyor
22          {
23              yeni = (dugum *) malloc(sizeof(dugum));
24              head = yeni;
25              tail = yeni;
26              yeni->sonraki = NULL;
27          }
28          else
29          {
30              yeni = (dugum *) malloc(sizeof(dugum));
31              yeni->onceki = tail;
32              tail->sonraki = yeni;
33              tail = yeni;
34              yeni->sonraki = NULL;
35          }
36          scanf("%d %s %d",&yeni->no,yeni->adi,&yeni->yas);
37      }
38  }
```

Traversing List

```
40  void listeDolas()
41  {
42      int sayac =1;
43      dugum *p;
44      p = head;
45      while(p!=NULL)
46      {
47          printf("%d- %d %s %d \n",sayac,p->no,p->adi,p->yas);
48          p = p->sonraki;
49          sayac++;
50      }
51  }
```

Inserting Node

```
53 void dugumEkle()
54 {
55     int kayitNo;
56     dugum *p,*yeniDugum;
57     yeniDugum = (dugum *) malloc(sizeof(dugum));
58     printf("Yeni dugum icin veri gir");
59     scanf("%d %s %d",&yeniDugum->no,yeniDugum->adi,&yeniDugum->yas);
60 
61     printf("Hangi kayittan oncesine eklemek istiyorsunuz");
62     printf("Liste sonuna eklemek icin 0 gir");
63     scanf("%d",&kayitNo);
64 
65     p = head;
66     if(p->no == kayitNo) //başa ekle
67     {
68         yeniDugum->sonraki = head;
69         head = yeniDugum;
70     }
```

Inserting Node

```
71      else
72      {
73          while(p->sonraki != NULL && p->no != kayitNo)
74          {
75              p= p->sonraki;
76          }
77          if(p->no == kayitNo) //Araya ekleme
78          {
79              yeniDugum->onceki = p->onceki;
80              p->onceki->sonraki = yeniDugum;
81              yeniDugum->sonraki = p;
82              p->onceki = yeniDugum;
83          }
84          else if(p->sonraki == NULL) //Sona ekleme
85          {
86              p->sonraki = yeniDugum;
87              yeniDugum->onceki = p;
88              yeniDugum->sonraki = NULL;
89              tail = yeniDugum;
90          }
91      }
92  }
```

Delete Node

```
 94  void dugumSil()
 95  {
 96      int kayitNo;
 97      dugum *p;
 98
 99      printf("Silmek istediginiz kayit no gir");
100      scanf("%d",&kayitNo);
101
102      p = head;
103      if(p->no == kayitNo) //baştakini sil
104      {
105          head->sonraki->onceki = NULL;
106          head = p->sonraki;
107          free(p);
108      }
```

Delete Node

```
109     else
110     {
111         while(p->sonraki != NULL && p->no != kayitNo)
112         {
113             p= p->sonraki;
114         }
115         if(p->no == kayitNo) //Aradan silme
116         {
117             p->onceki->sonraki = p->sonraki;
118             if(p!=tail)
119                 p->sonraki->onceki = p->onceki;
120             if(p==tail) tail = p->onceki;
121             free(p);
122         }
123         else if(p->sonraki == NULL) //Silinecek düğüm bulunamadı
124         {
125             printf("Silinecek dugum bulunamadi");
126         }
127     }
128 }
```

Doubly Linked List Application

```
130 int main(void)
131 {
132     int secim=0;
133     head = NULL, tail = NULL, yeni = NULL;
134     printf("1-Liste Olustur 2-Liste Dolas 3-Dugum Sil 4-Dugum Ekle 5-Cikis\n");
135     while(1)
136     {
137         printf("Secim yap [1-5]?");
138         scanf("%d",&secim);
139         switch(secim)
140         {
141         case 1: listeOlustur();
142             listeDolas();break;
143         case 2: listeDolas();break;
144         case 3: dugumSil();
145             listeDolas();break;
146         case 4: dugumEkle();
147             listeDolas();break;
148         case 5: exit(0);
149         }
150     }
151 }
```