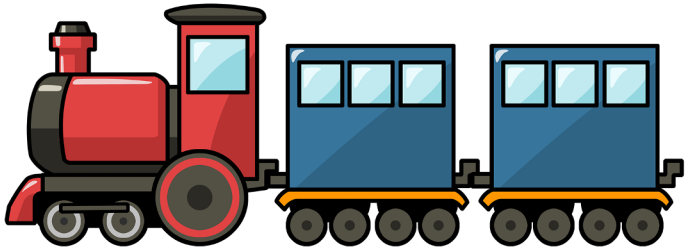
Baglantılı Listeler
Linked List



Suhap SAHIN
Onur GÖK

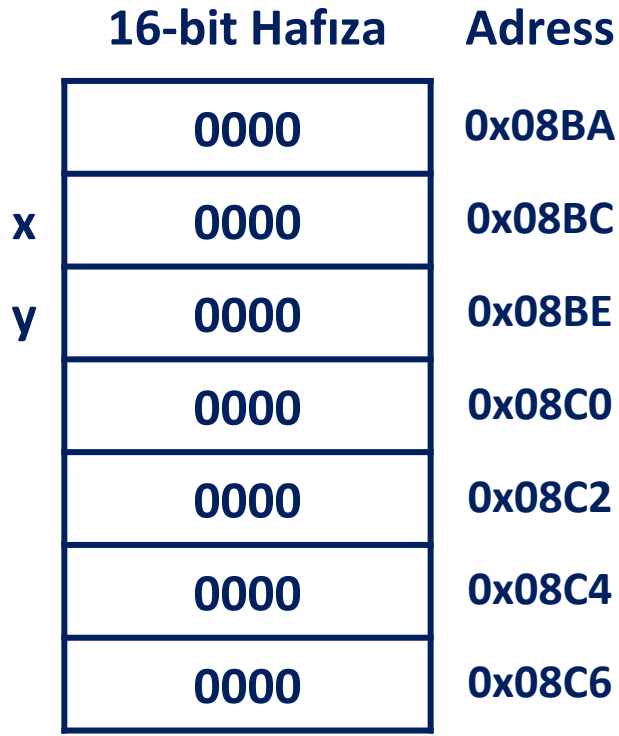
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```

16-bit Hafıza	Adress
0000	0x08BA
0000	0x08BC
0000	0x08BE
0000	0x08C0
0000	0x08C2
0000	0x08C4
0000	0x08C6

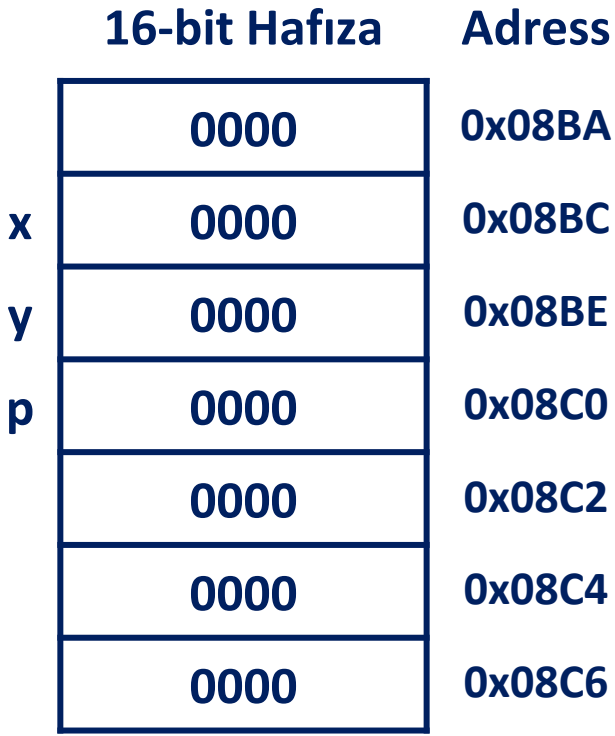
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



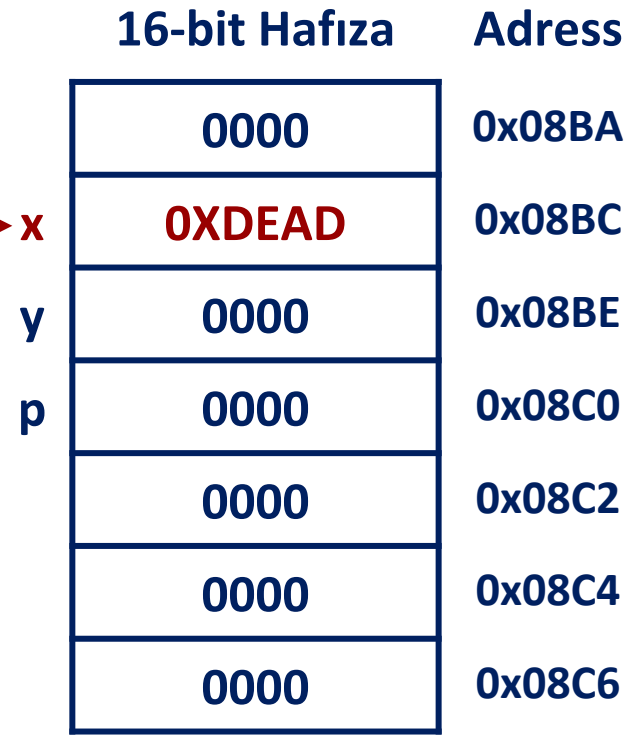
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



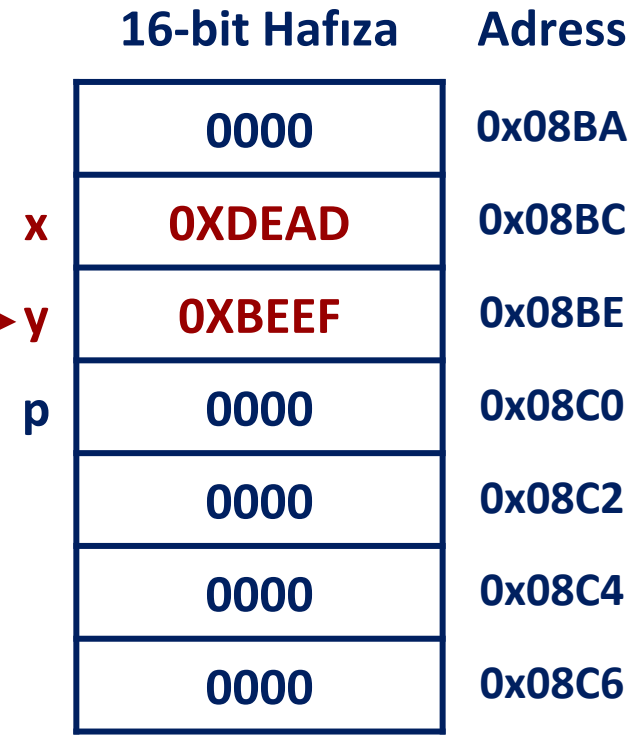
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



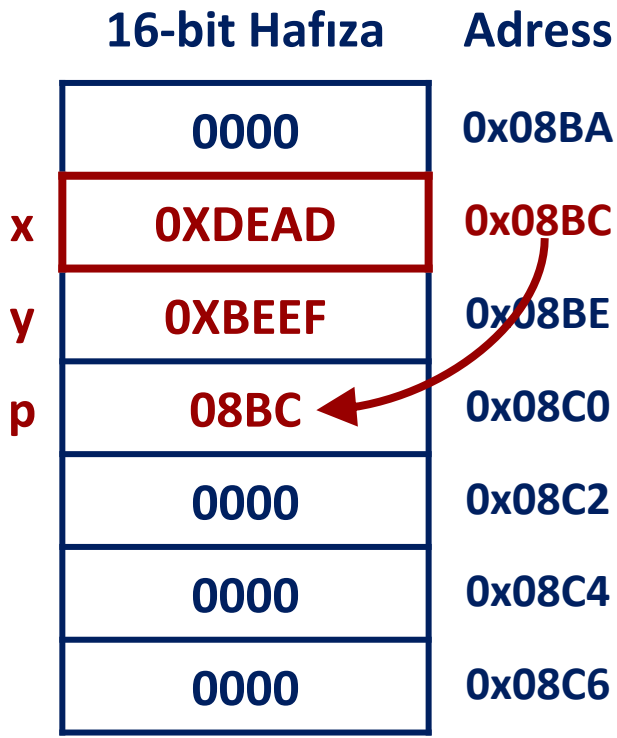
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



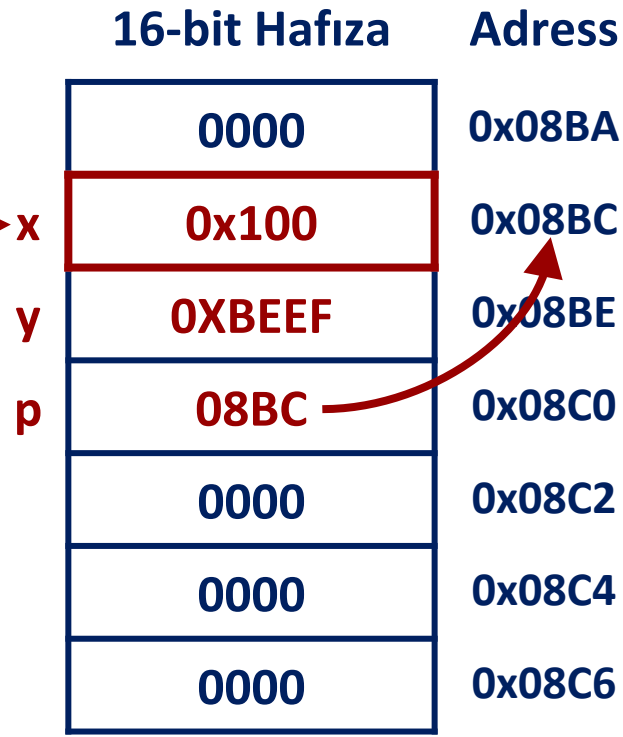
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



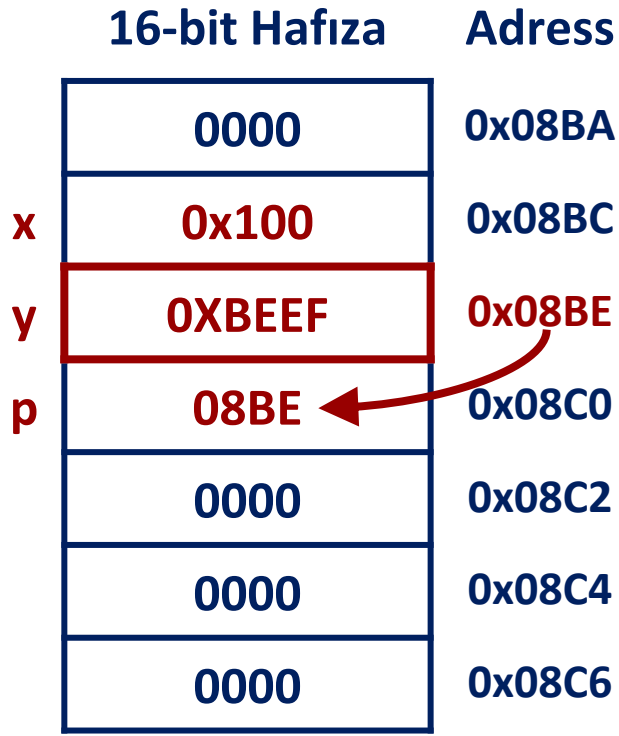
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
       *p = 0x200;
6      return 0;
    }
7
```



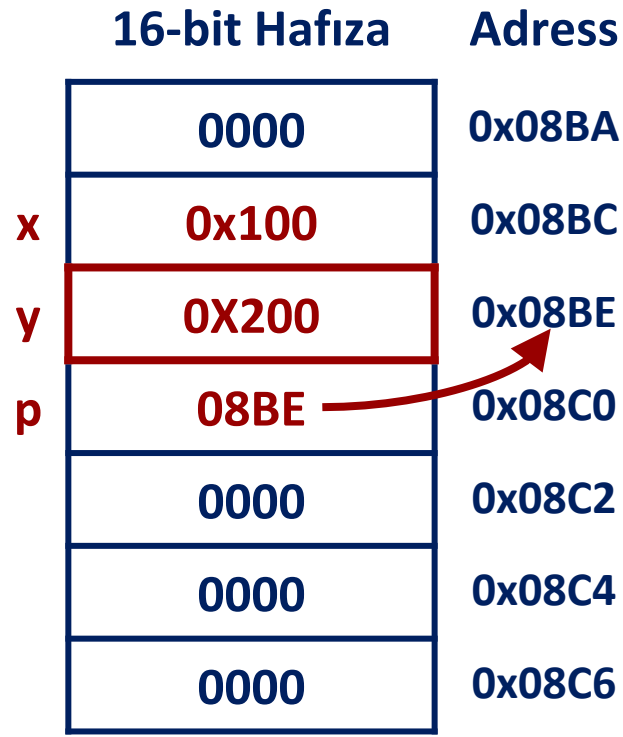
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```



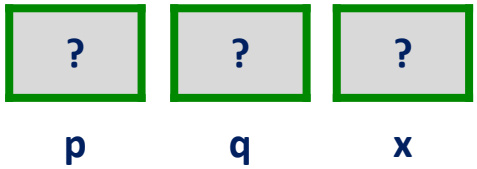
isaretciler (Pointers)

```
    #include <stdio.h>
1   int main(void)
    {
2       int x,y;
        int *p;
3        x = 0xDEAD;
         y = 0xBEEF;
4        p = &x;
        *p = 0x100;
5        p = &y;
        *p = 0x200;
6       return 0;
    }
7
```

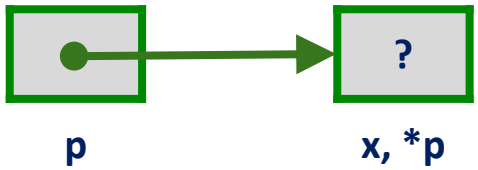


isaretciler (Pointers)

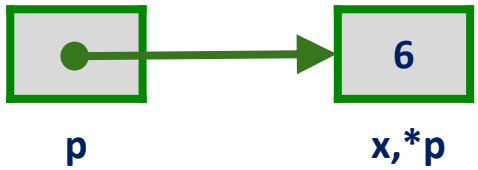
int *p,*q;
int x;



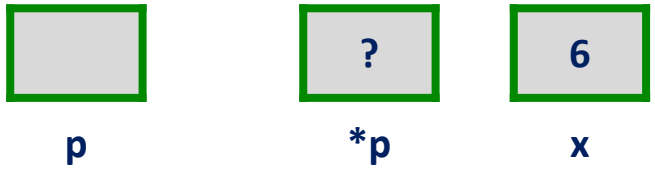
p = &x;



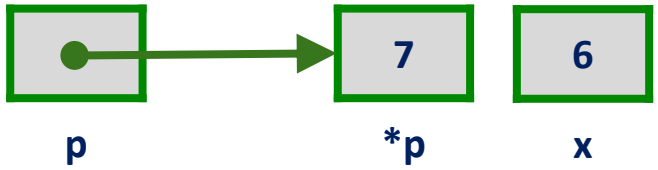
*p = 6;



p = **new** int;

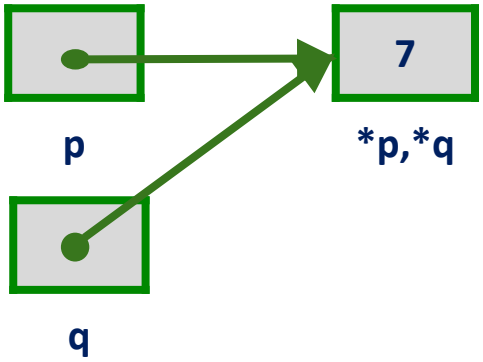


*p = 7;

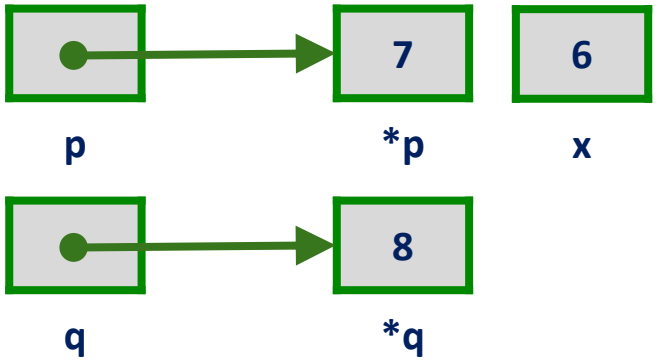


isaretciler (Pointers)

q = p;

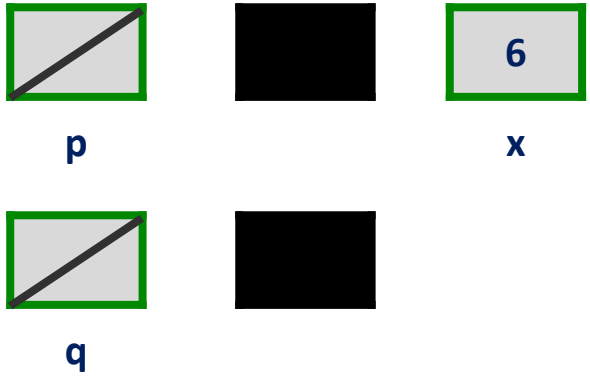


q = **new** int;
*q = 8;



p = NULL;

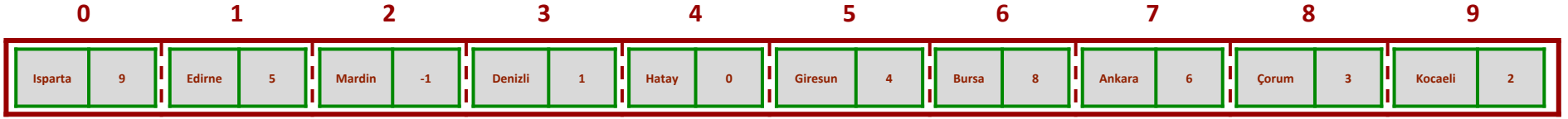
Delete q;
q = NULL;



il sıralaması

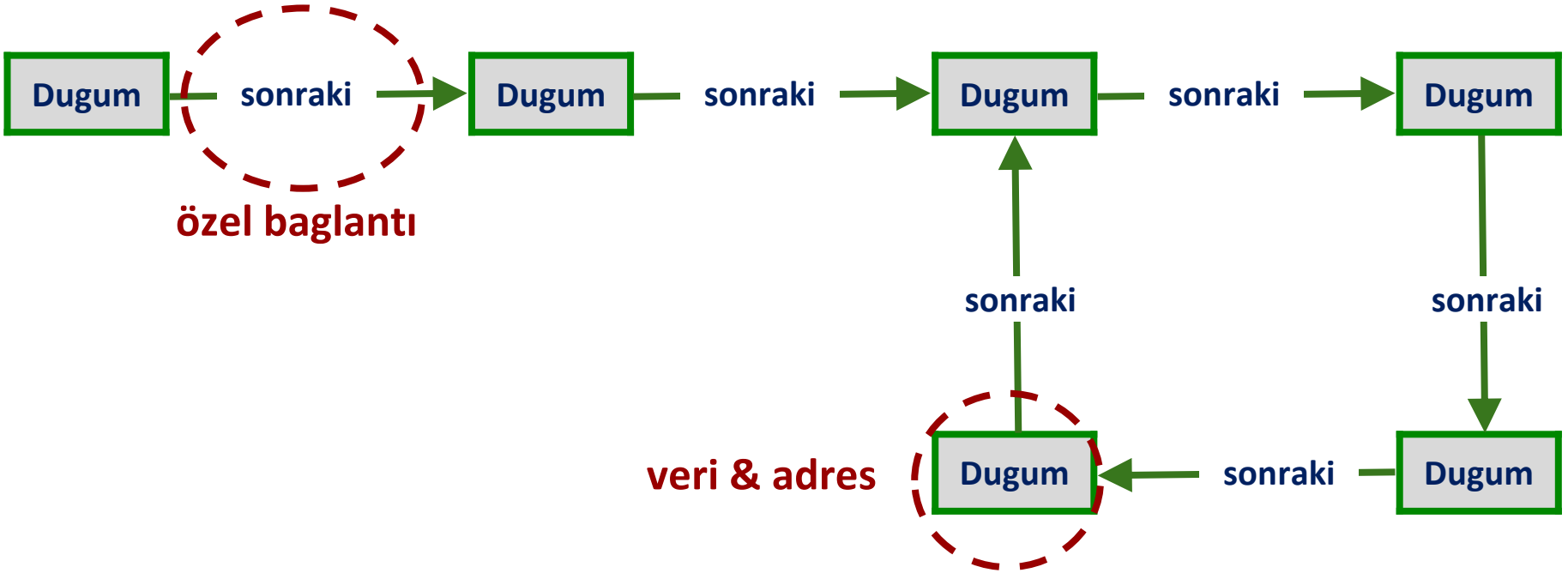

```
struct Dugum
{
  string il;
  int   sonraki;
};
```

il sıralaması



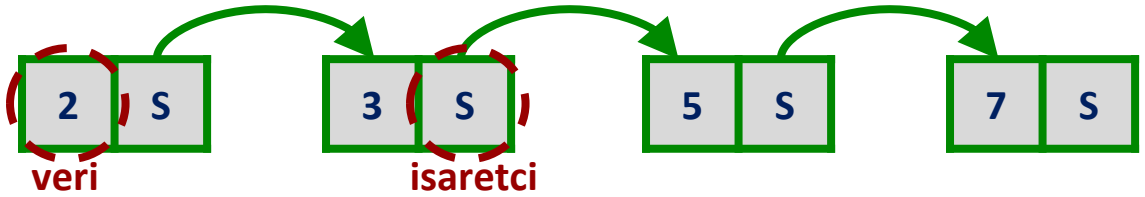
Adress	Bellek icerigi
0F20	Isparta | 9
0F24	Edirne | 5
0F28	Mardin | -1
0F2C	Denizli | 1
0F30	Hatay | 0
0F34	Giresun | 4
0F38	Bursa | 8
0F3C	Ankara | 6
0F40	Çorum | 3
0F44	Kocaeli | 2

Baglantılı listeler



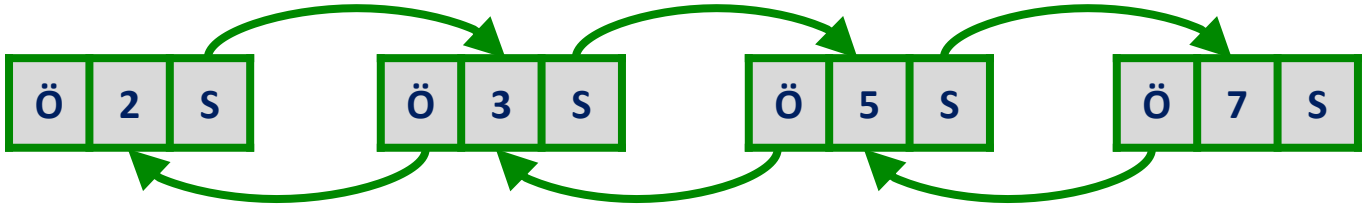
Baglantılı listeler

Tek yönlü baglantılı liste



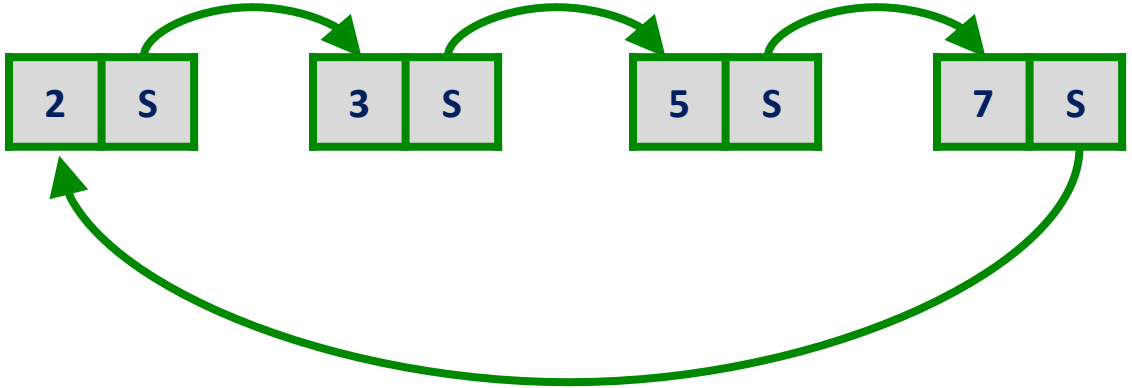
Baglantılı listeler

Çift yönlü baglantılı liste



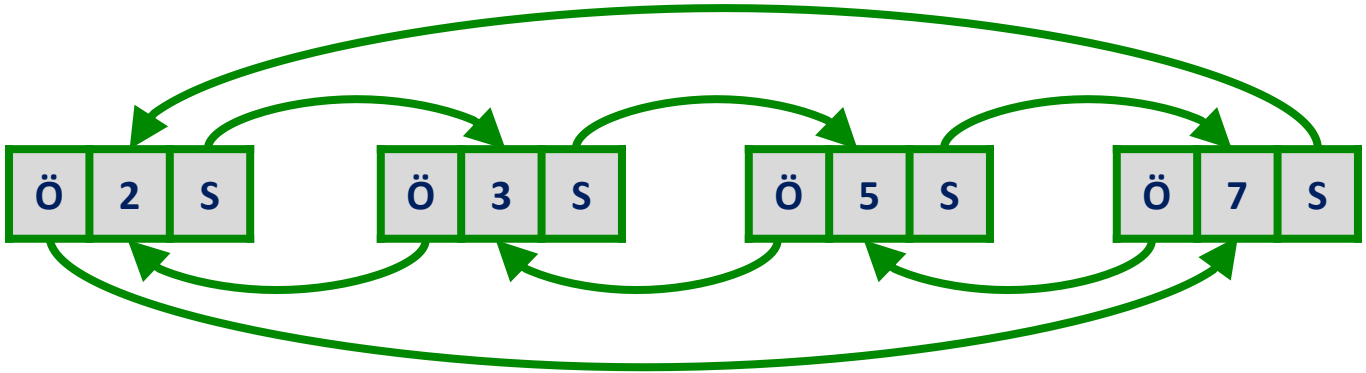
Baglantılı listeler

Tek yönlü çevrimsel baglantılı liste



Baglantılı listeler

Çift yönlü çevrimsel baglantılı liste



Dügüm(Node)

C & C++

Dügüm

string veri;	int sonraki;

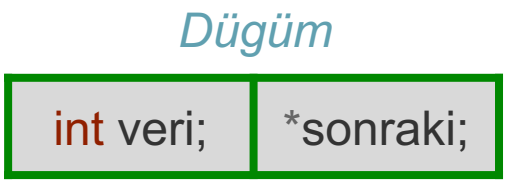
```
// Dügüm
struct Dugum
{
  string veri;
  int   sonraki;
};
```

il sıralaması

Dinamik bellek kullanımı
&
Nesneye yönelik programlama

Dügüm(Node)

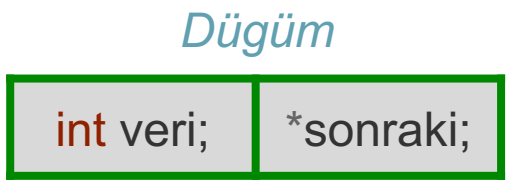
C & C++




```
// Dügüm
struct Dugum
{
  int veri;
  struct Dugum *sonraki;
};
```

Tek Yönlü Baglantılı Liste

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



```
struct Dugum* birinci = NULL;
struct Dugum* ikinci  = NULL;
struct Dugum* ucuncu  = NULL;
```

Tek Yönlü Baglantılı Liste

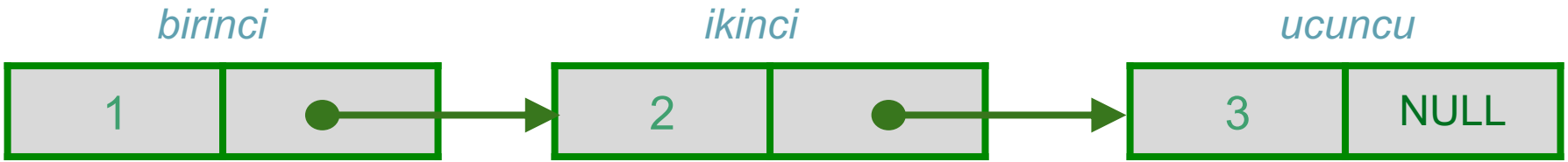
```
birinci = (struct Dugum*)malloc(sizeof(struct Dugum));
ikinci = (struct Dugum*)malloc(sizeof(struct Dugum));
ucuncu = (struct Dugum*)malloc(sizeof(struct Dugum));
```

Tek Yönlü Baglantılı Liste

```
birinci->veri = 1;
birinci->sonraki = ikinci;
```

```
ikinci->veri = 2;
ikinci->sonraki = ucuncu;
```

```
ucuncu->veri = 3;
ucuncu->sonraki = NULL;
```

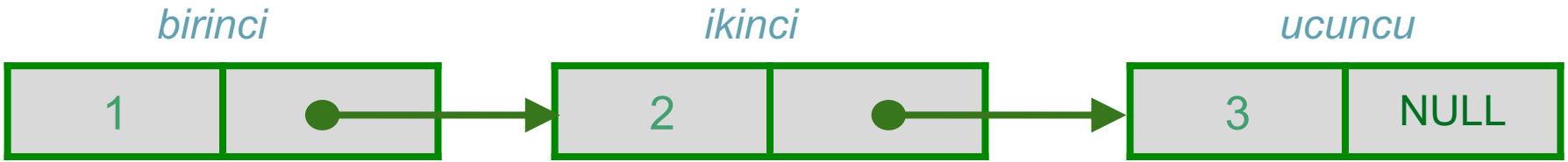


Tek Yönlü Baglantılı Liste

```
birinci->veri = 1;
birinci->sonraki = ikinci;
```

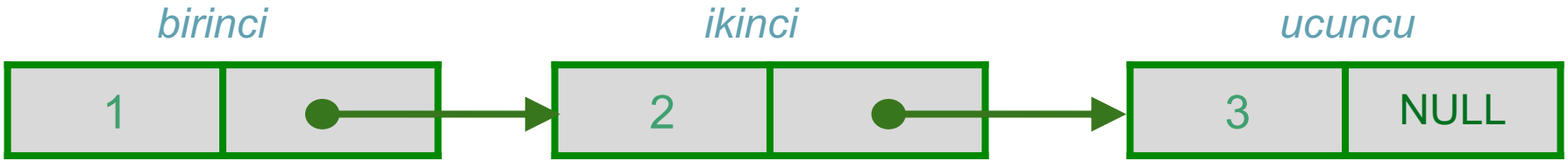
```
ikinci->veri = 2;
ikinci->sonraki = ucuncu;
```

```
ucuncu->veri = 3;
ucuncu->sonraki = NULL;
```



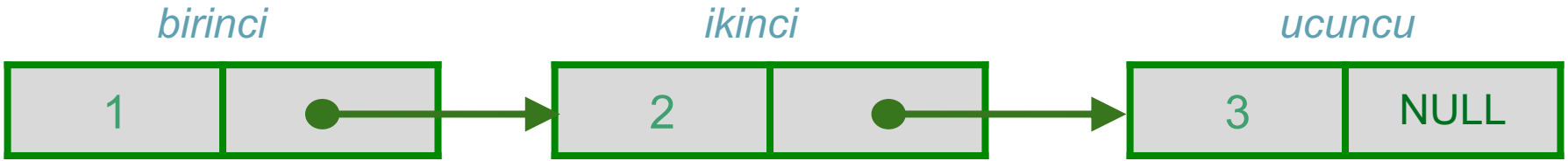
Tek Yönlü Baglantılı

```
void printList(struct Dugum *n){
  while (n != NULL){
     printf(" %d ", n->veri);
     n = n->sonraki;
  }
}
```



Tek Yönlü Baglantılı Liste

```
void printList(struct Dugum *n){
        printf(" %d ", n->veri);
    printf(" %d ", n->sonraki->veri);
    printf(" %d ", n->sonraki->sonraki->veri);
}
```



Ekleme

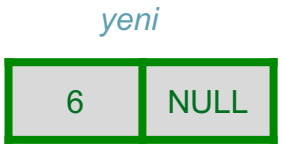
```
#include <stdio.h>
#include <stdlib.h>

struct Dugum{
        int veri;
        struct Dugum *sonraki;
};

int main(){
        struct Dugum* yeni = NULL;



        return 0;
}
```



Sona Ekleme

```
/* sona ekleme  */
void sona_ekle(struct Dugum** yeni_ref, int yeni_veri)
{
  /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   struct Dugum *sonuncu = *yeni_ref;
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   yeni_dugum->sonraki = NULL;









   return;
}
```

```
sona_ekle(&yeni, 6);
```

yeni

Sona Ekleme

```
/* sona ekleme  */
void sona_ekle(struct Dugum** yeni_ref, int yeni_veri)
{
  /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   struct Dugum *sonuncu = *yeni_ref;
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   yeni_dugum->sonraki = NULL;
  /* 3. Eğer liste boşsa */
   if (*yeni_ref == NULL)
   {
     *yeni_ref = yeni_dugum;
     return;
   }




   return;
}
```

```
sona_ekle(&yeni, 6);
```

yeni

Sona Ekleme

```
/* sona ekleme  */
void sona_ekle(struct Dugum** yeni_ref, int yeni_veri)
{
  /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   struct Dugum *sonuncu = *yeni_ref;
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   yeni_dugum->sonraki = NULL;
  /* 3. Eğer liste boşsa */
   if (*yeni_ref == NULL)
   {
     *yeni_ref = yeni_dugum;
     return;
   }
   /* 4. Boş değilse son düğüme kadar git */
   while (sonuncu->sonraki != NULL)
      sonuncu = sonuncu->sonraki;


   return;
}
```

```
sona_ekle(&yeni, 6);
```

yeni

Sona Ekleme

```
/* sona ekleme  */
void sona_ekle(struct Dugum** yeni_ref, int yeni_veri)
{
  /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   struct Dugum *sonuncu = *yeni_ref;
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   yeni_dugum->sonraki = NULL;
  /* 3. Eğer liste boşsa */
   if (*yeni_ref == NULL)
   {
     *yeni_ref = yeni_dugum;
     return;
   }
   /* 4. Boş değilse son düğüme kadar git */
   while (sonuncu->sonraki != NULL)
      sonuncu = sonuncu->sonraki;
   /* 5. sonuncuya kadar gittik */
   sonuncu->sonraki = yeni_dugum;
   return;
}
```

```
sona_ekle(&yeni, 6);
```

yeni

Basa Ekleme

```
basa_ekle(&yeni, 7);
```

```
void basa_ekle(struct Dugum** birinci_ref, int yeni_veri)
{
   /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 3. Yeni dugum birinci dugum olmalı */
   yeni_dugum->sonraki = (*birinci_ref);
   (*birinci_ref)    = yeni_dugum;
}
```

birinci_ref

yeni_dugum

Basa Ekleme

```
basa_ekle(&yeni, 7);
```

```
void basa_ekle(struct Dugum** birinci_ref, int yeni_veri)
{
   /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 3. Yeni dugum birinci dugum olmalı */
   yeni_dugum->sonraki = (*birinci_ref);
   (*birinci_ref)    = yeni_dugum;
}
```

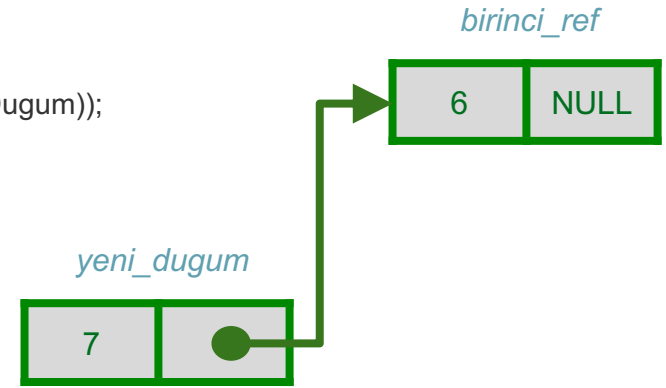
birinci_ref

yeni_dugum

Basa Ekleme

```
basa_ekle(&yeni, 7);
```

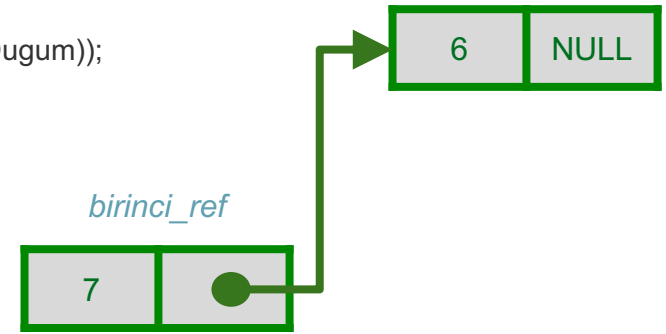
```
void basa_ekle(struct Dugum** birinci_ref, int yeni_veri)
{
   /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 3. Yeni dugum birinci dugum olmalı */
   yeni_dugum->sonraki = (*birinci_ref);
   (*birinci_ref)    = yeni_dugum;
}
```



Basa Ekleme

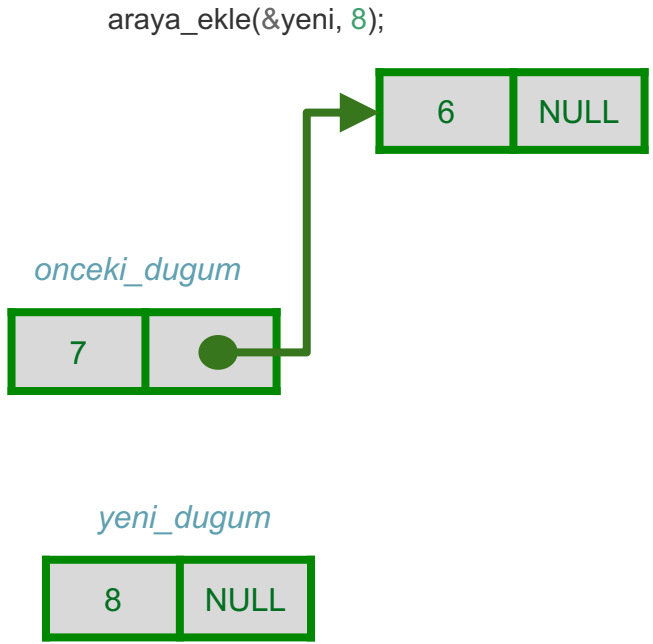
```
basa_ekle(&yeni, 7);
```

```
void basa_ekle(struct Dugum** birinci_ref, int yeni_veri)
{
   /* 1. yeni dugum için yer al */
   struct Dugum* yeni_dugum = (struct Dugum*) malloc(sizeof(struct Dugum));
   /* 2. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 3. Yeni dugum birinci dugum olmalı */
   yeni_dugum->sonraki = (*birinci_ref);
   (*birinci_ref)    = yeni_dugum;
}
```



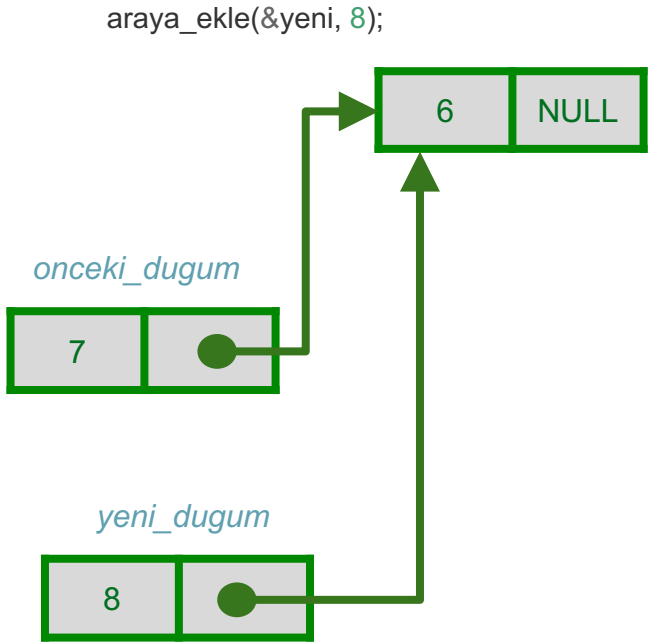
Araya Ekleme

```
void araya_ekle(struct Dugum* onceki_dugum, int yeni_veri)
{
   /*1. onceki dugum NULL mu? */
   if (onceki_dugum == NULL)
   {
     printf("onceki dugum NULL olmamalı");
     return;
   }
   /* 2. yeni dugum için yer al */
   struct Dugum* yeni_dugum =(struct Dugum*) malloc(sizeof(struct Dugum));
   /* 3. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 4. yeni dugum araya al */
   yeni_dugum->sonraki = onceki_dugum->sonraki;
   onceki_dugum->sonraki = yeni_dugum;
}
```



Araya Ekleme

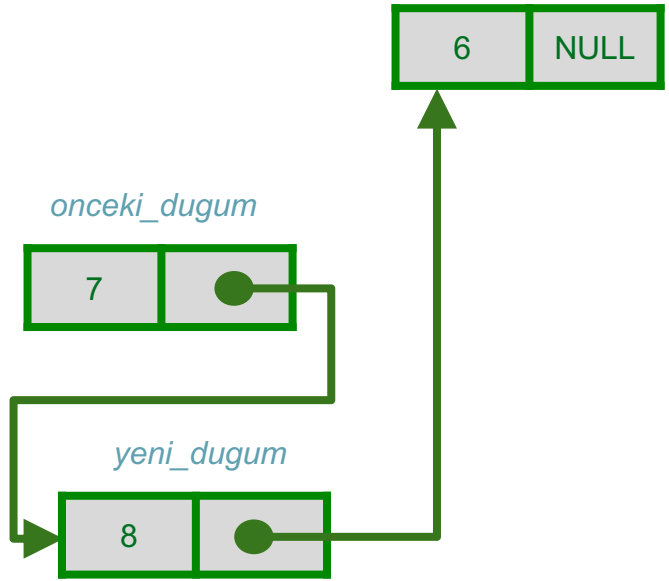
```
void araya_ekle(struct Dugum* onceki_dugum, int yeni_veri)
{
   /*1. onceki dugum NULL mu? */
   if (onceki_dugum == NULL)
   {
     printf("onceki dugum NULL olmamalı");
     return;
   }
   /* 2. yeni dugum için yer al */
   struct Dugum* yeni_dugum =(struct Dugum*) malloc(sizeof(struct Dugum));
   /* 3. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 4. yeni dugum araya al */
   yeni_dugum->sonraki = onceki_dugum->sonraki;
   onceki_dugum->sonraki = yeni_dugum;
}
```



Araya Ekleme

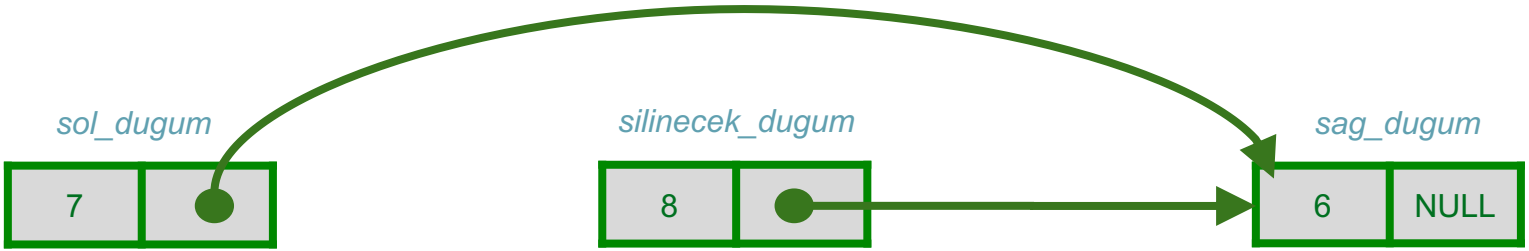
```
void araya_ekle(struct Dugum* onceki_dugum, int yeni_veri)
{
   /*1. onceki dugum NULL mu? */
   if (onceki_dugum == NULL)
   {
     printf("onceki dugum NULL olmamalı");
     return;
   }
   /* 2. yeni dugum için yer al */
   struct Dugum* yeni_dugum =(struct Dugum*) malloc(sizeof(struct Dugum));
   /* 3. yeni dugume yeni veri  */
   yeni_dugum->veri  = yeni_veri;
   /* 4. yeni dugum araya al */
   yeni_dugum->sonraki = onceki_dugum->sonraki;
   onceki_dugum->sonraki = yeni_dugum;
}
```

```
araya_ekle(&yeni, 8);
```

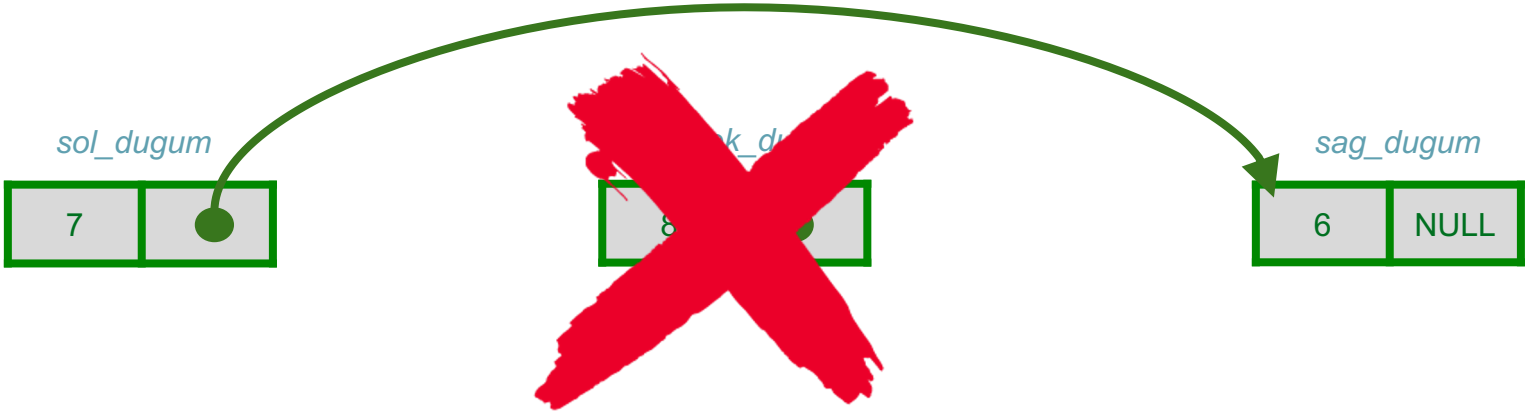


Silme

Silme



Silme



Silme

Sorular