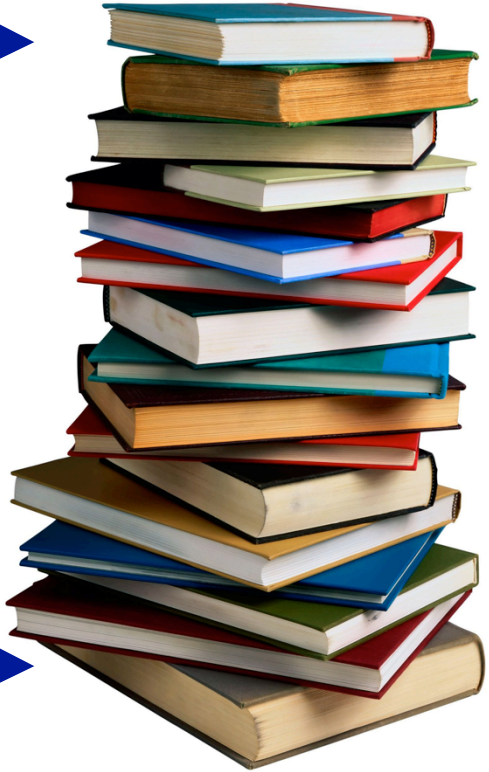
Yıgın & Kuyruk

Suhap
SAHIN
Onur GÖK

Yığın (Stack)

son giren

ilk çıkan

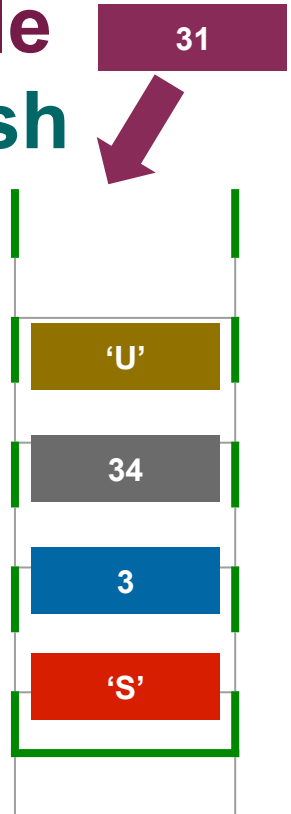


ilk giren

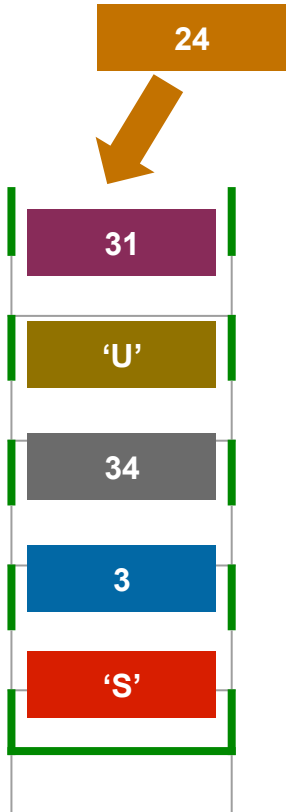
son çıkan

Yıgın islemleri(SO)

ekle
push



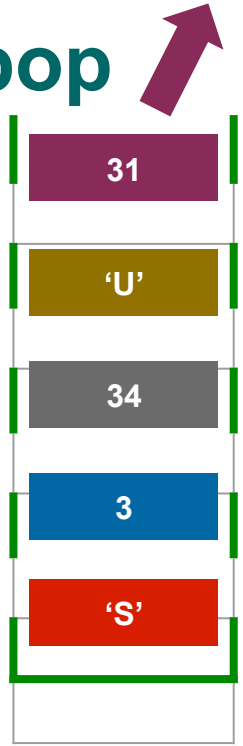
stack
overflow



Yıgın islemleri(SO)

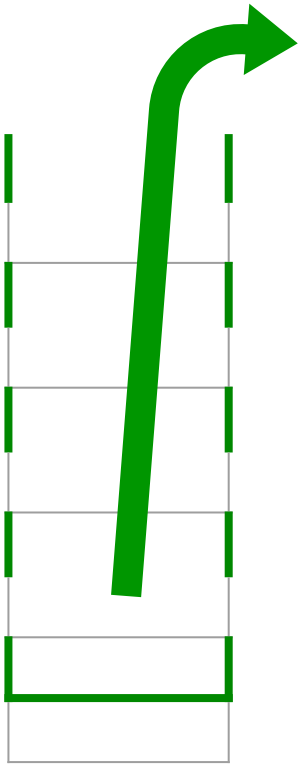
al

pop

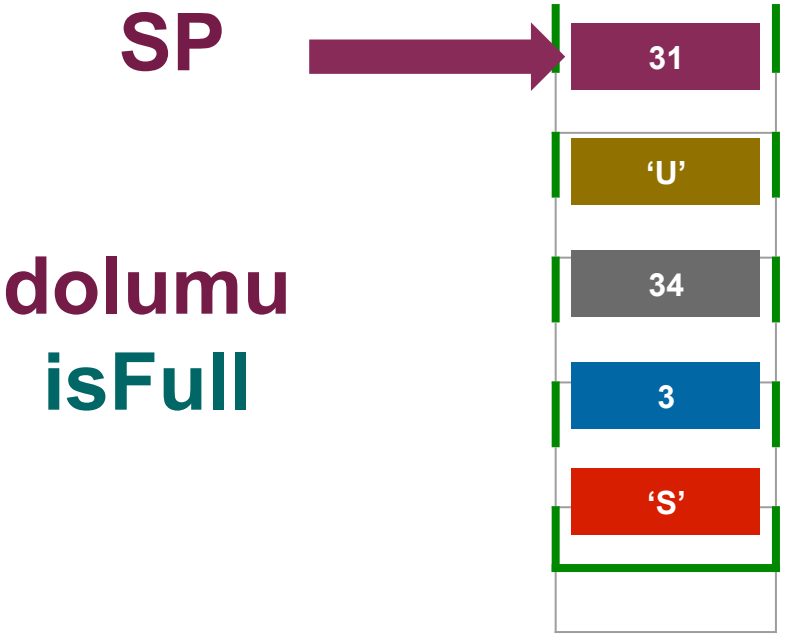


stack

underflow



Yıgın islemleri(SO)

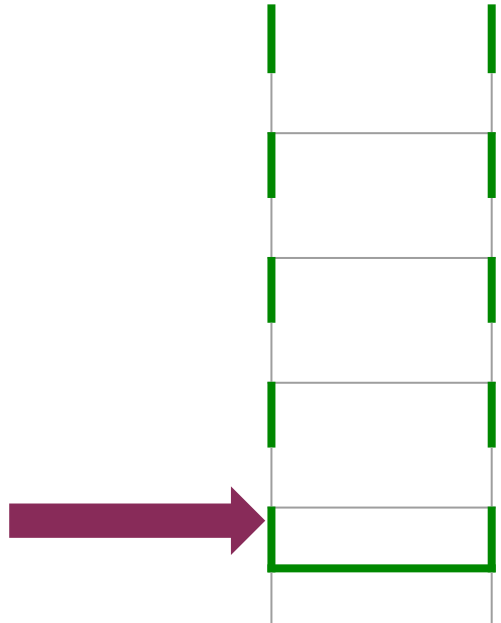


Yıgın islemleri(SO)

bosmu

isEmpty

SP



Yığın islemleri(SO)

ekle(), push()
al(), pop()
bosmu(), isEmpty()
dolumu(), isFull()

O(1)

Yığın Uygulamaları

Sembollerin esitlenmesi

```
#include <stdio.h>
int main () {
    for(int i=0;i<10;i++){
        // Kod
    }
}
```

Yığın Uygulamaları

Son-ek(Posfix)/Ön-ek(Prefix) ifadeleri

a op1 b op2 c op3 d

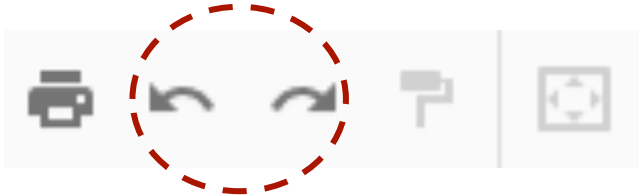
op1 = +

op2 = *

op3 = +

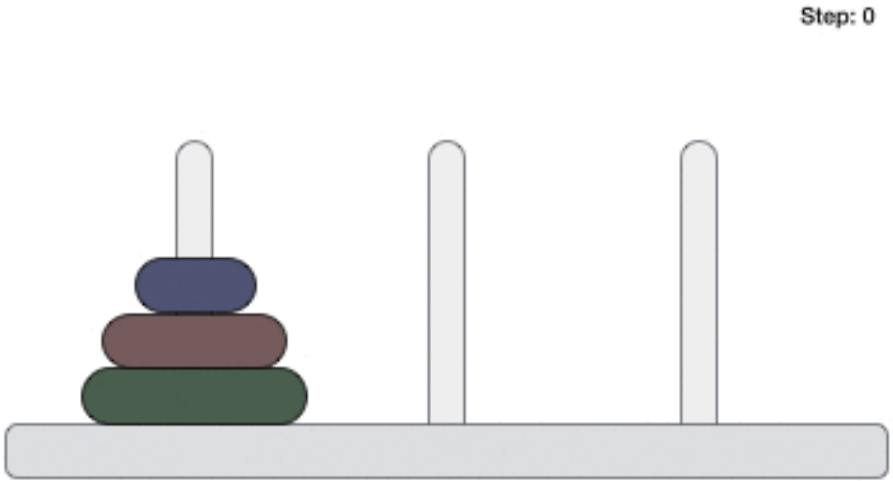
Yığın Uygulamaları

Bir çok programdaki ileri-al, geri-al (redo-undo) ve ileri-git, geri-git (forward-backward) özellikleri



Yığın Uygulamaları

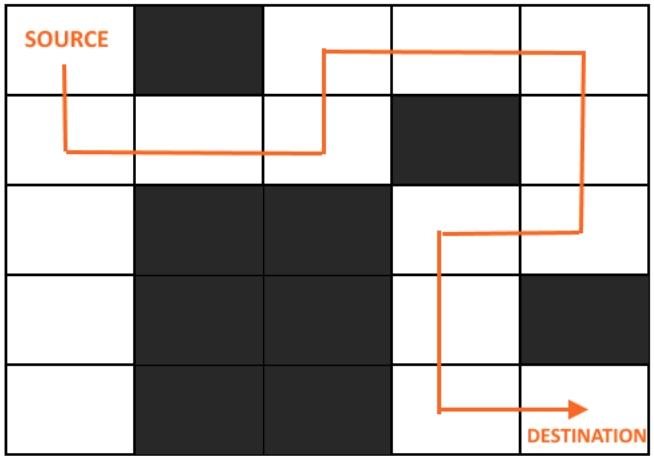
Tower of Hanoi, tree traversals ...



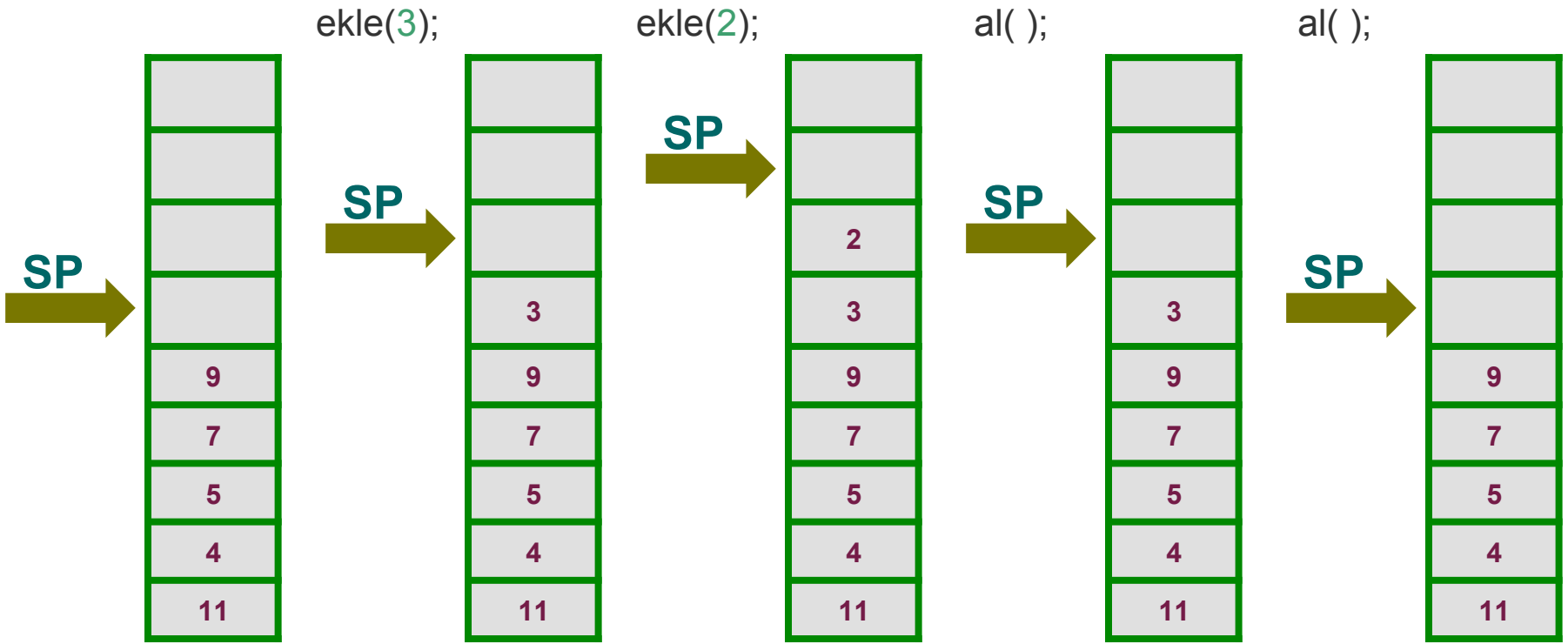
Yığın Uygulamaları

Backtracking, Knight tour problem, rat in a maze, …

sudoku solver

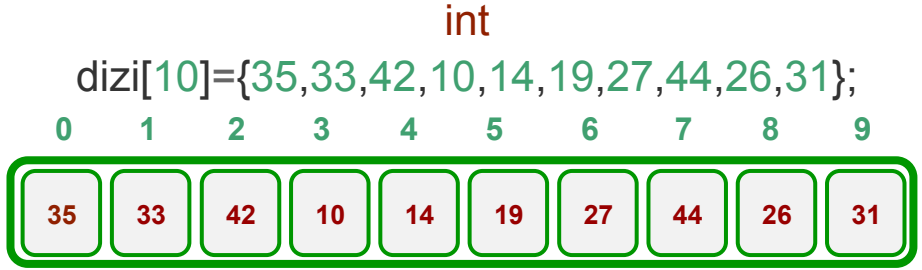


Yığın isaretcisi (SP)

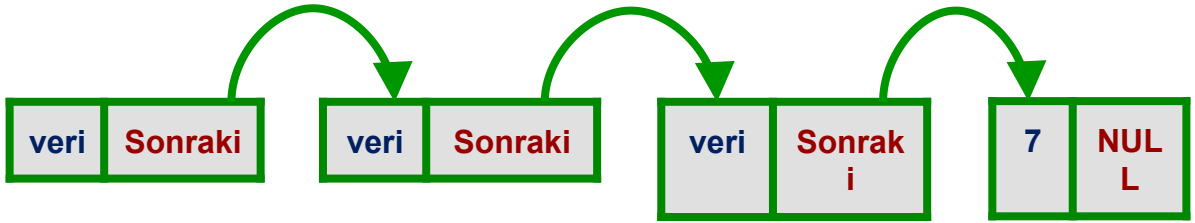


Yıgın Gerçeklestirimi

Dizi

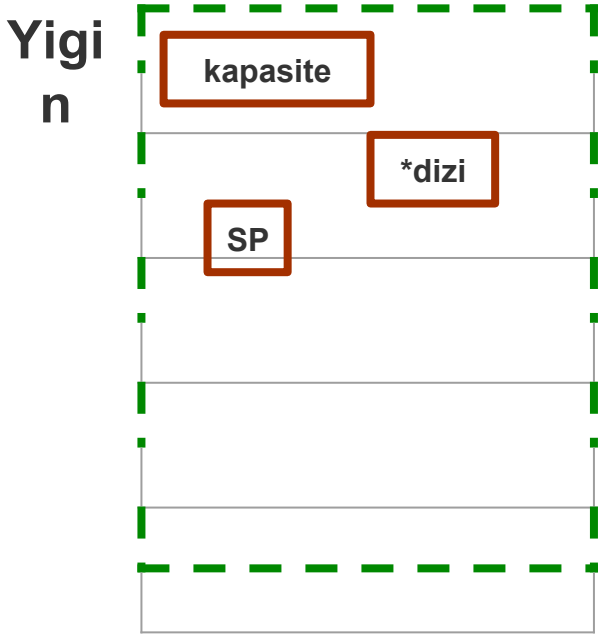


Baglantılı Listeler



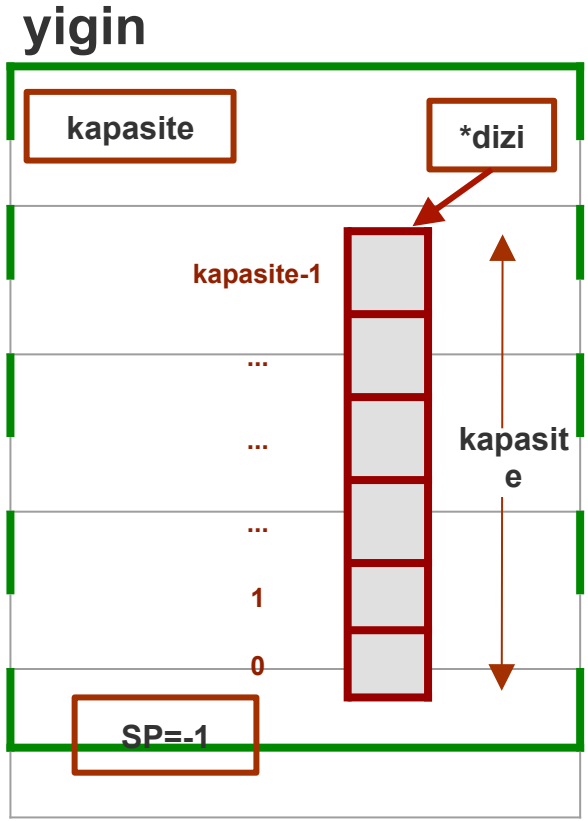
Yıgın(Dizi) & C

```
struct Yigin
{
    int SP;
    unsigned kapasite;
    int* dizi;
};
```



Yıgın(Dizi) & C

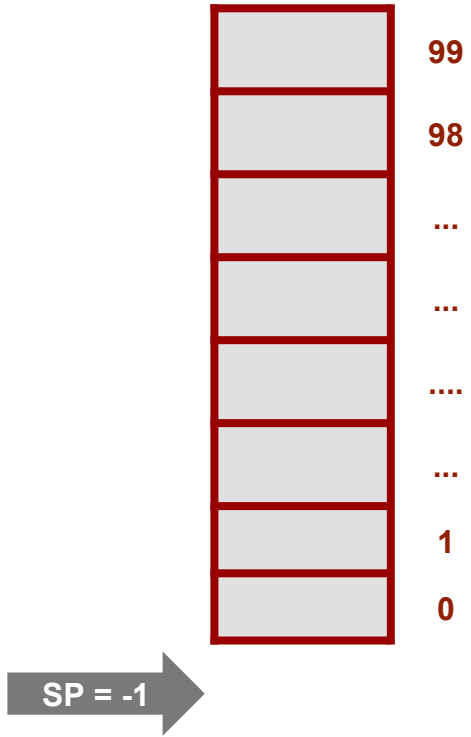
```
struct Yigin* yiginOlustur(unsigned kapasite)
{
    struct Yigin* yigin = (struct Yigin*) malloc(sizeof(struct Yigin));
    yigin->kapasite = kapasite;
    yigin->SP = -1;
    yigin->dizi = (int*) malloc(yigin->kapasite * sizeof(int));
    return yigin;
}
```



Yıgın(Dizi) & C

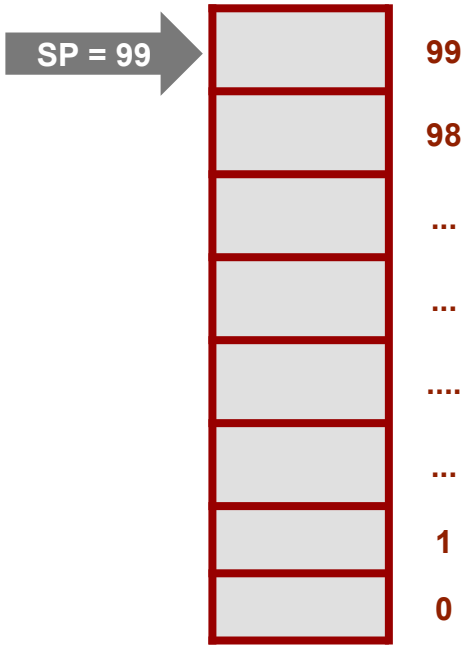
```
struct Yigin* yiginOlustur(unsigned kapasite)
{
    struct Yigin* yigin = (struct Yigin*) malloc(sizeof(struct Yigin));
    yigin->kapasite = kapasite;
    yigin->SP = -1;
    yigin->dizi = (int*) malloc(yigin->kapasite * sizeof(int));
    return yigin;
}
```

```
struct Yigin* yigin = yiginOlustur(100);
```



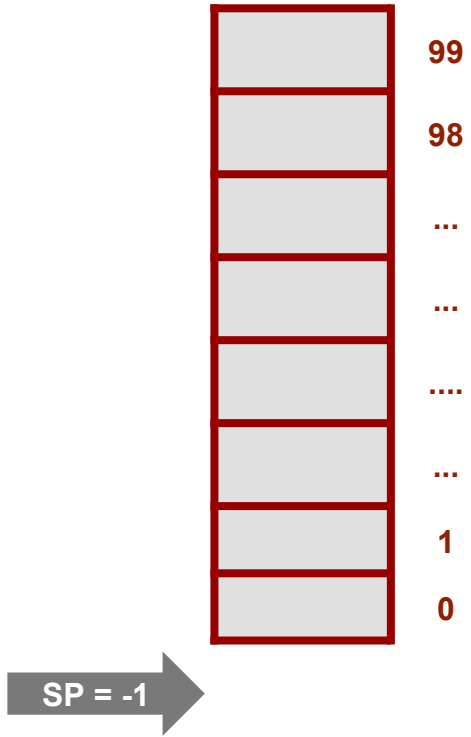
Yıgın(Dizi) & C

```
int dolumu(struct Yigin* yigin)
{   return yigin->SP == yigin->kapasite - 1; }
```



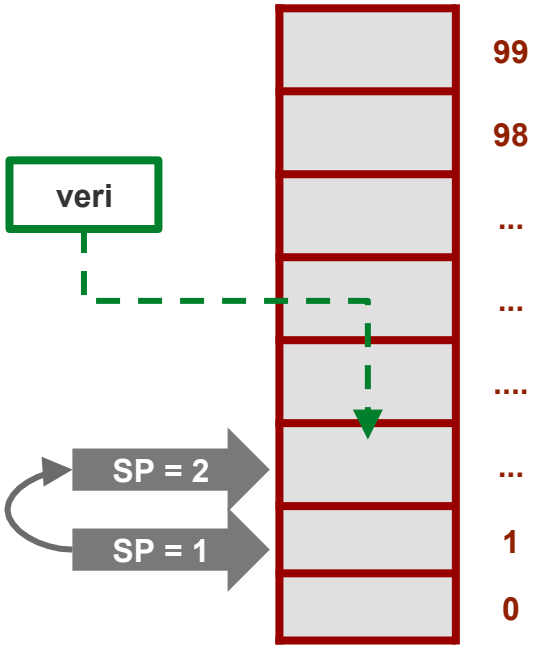
Yıgın(Dizi) & C

```
int bosmu(struct Yigin* yigin)
{   return yigin->SP == -1;  }
```



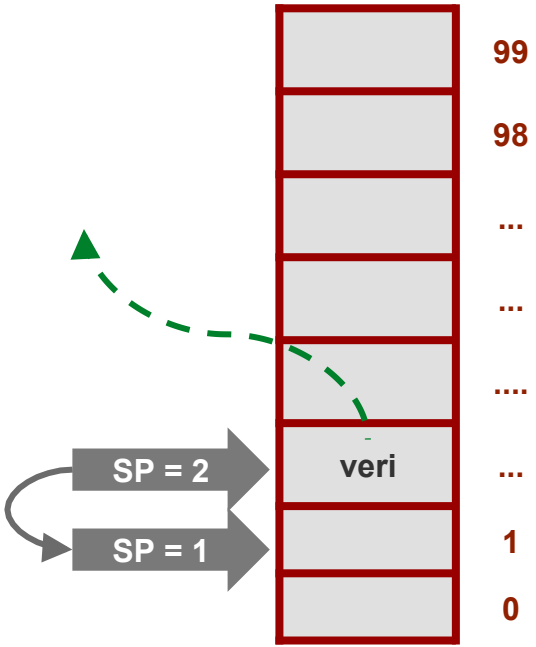
Yıgın(Dizi) & C

```
void ekle(struct Yigin* yigin, int veri)
{
    if (dolumu(yigin))
        return;
    yigin->dizi[++yigin->SP] = veri;
    printf("%d Yigina eklendi\n", veri);
}


ekle(yigin, 10);
```



Yıgın(Dizi) & C

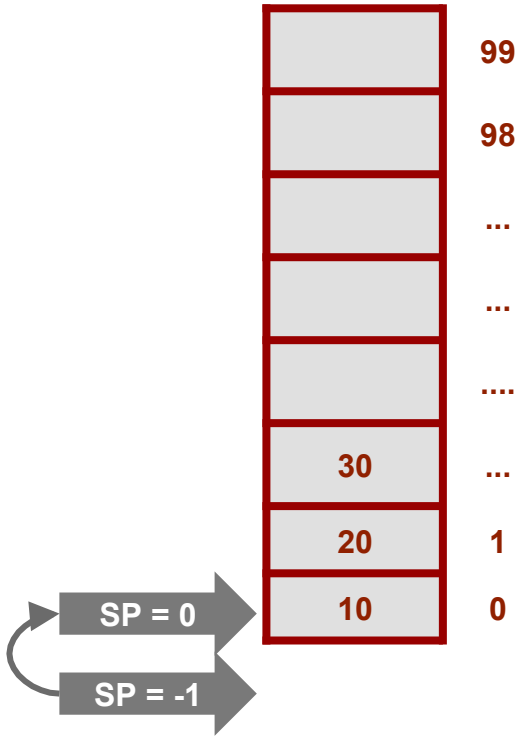
```
int al(struct Yigin* yigin)
{
    if (bosmu(yigin))
        return INT_MIN;
    return yigin->dizi[yigin->SP--];
}
```




```
al(yigin);
```

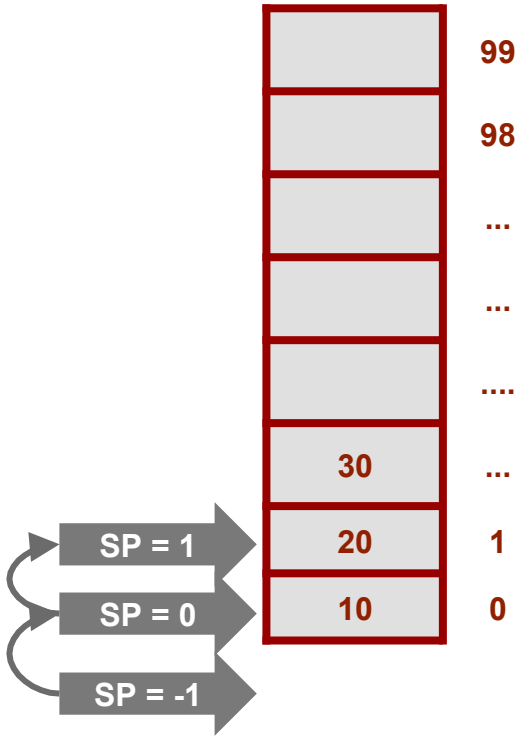
Yıgın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);









    return 0;
}
```



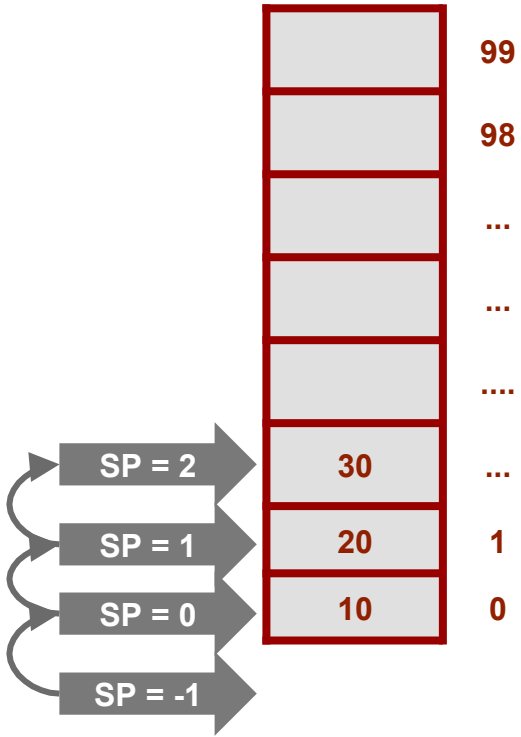
Yıgın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);









    return 0;
}
```



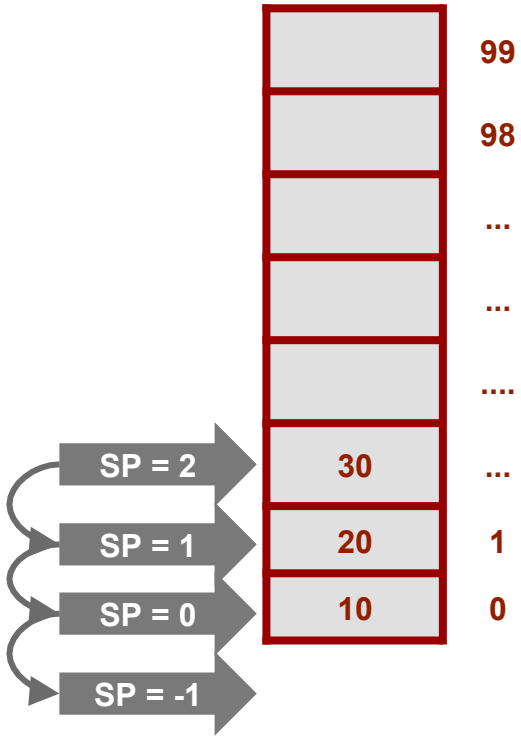
Yığın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);
    ekle(yigin, 30);




    return 0;
}
```



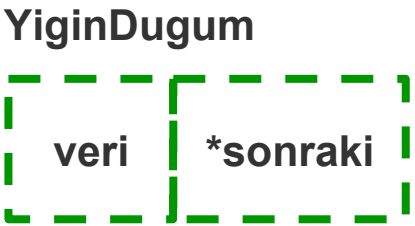
Yıgın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);
    ekle(yigin, 30);
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    return 0;
}
```



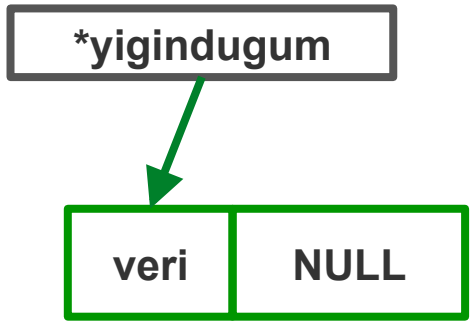
Yıgın(Baglı Liste) & C

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



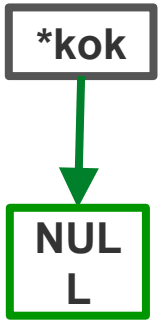
Yıgın(Baglı Liste) & C

```
struct YiginDugum* YeniDugum(int veri)
{
    struct YiginDugum* yigindugum = (struct YiginDugum*) malloc(sizeof(struct YiginDugum));
    yigindugum->veri = veri;
    yigindugum->sonraki = NULL;
    return yigindugum;
}
```



Yıgın(Baglı Liste) & C

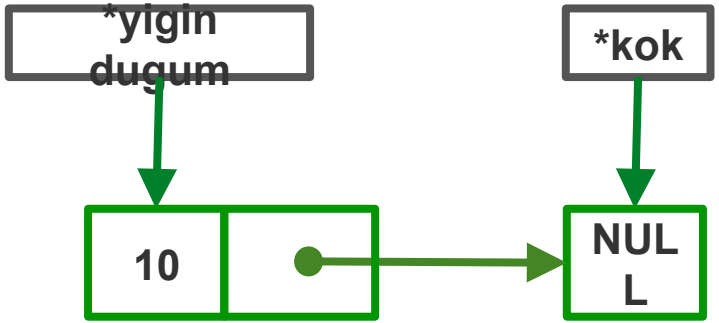
```
struct YiginDugum* kok = NULL;
```



Yıgın(Baglı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

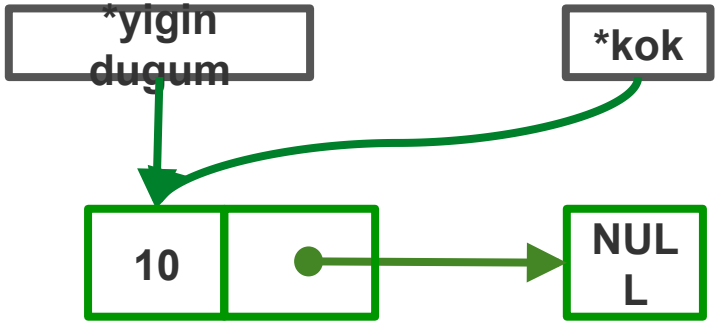
```
ekle(&kok, 10);
```



Yıgın(Baglı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

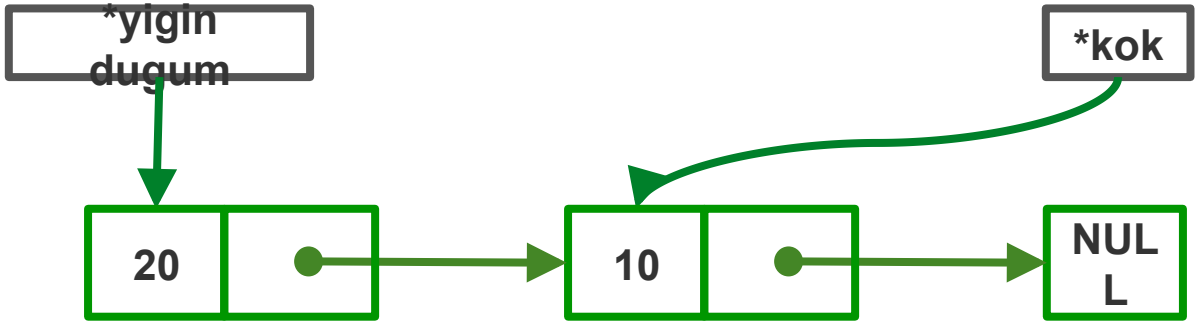
```
ekle(&kok, 10);
```



Yıgın(Baglı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

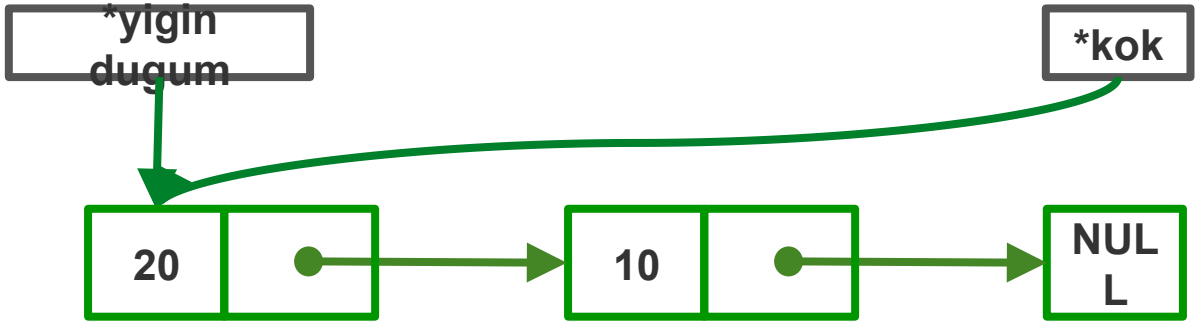
```
ekle(&kok, 10);

ekle(&kok, 20);
```



Yıgın(Baglı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

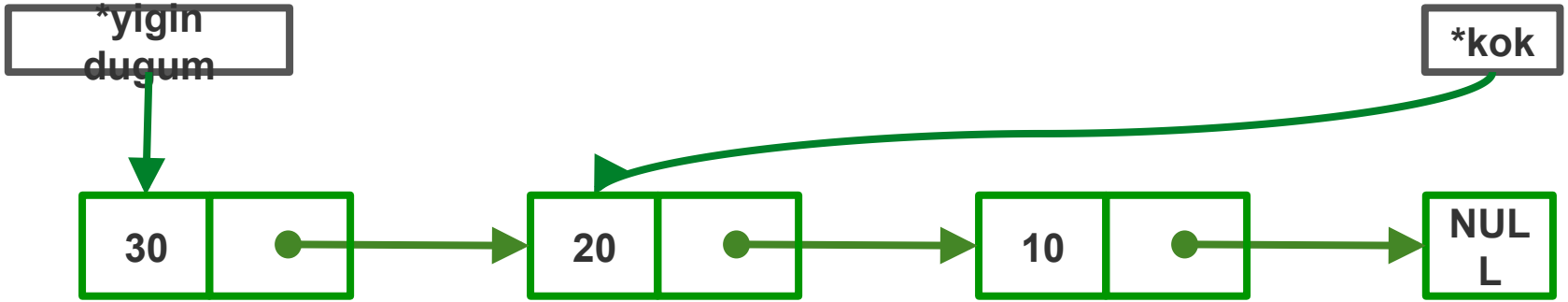
```
ekle(&kok, 10);

ekle(&kok, 20);
```



Yıgın(Baglı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

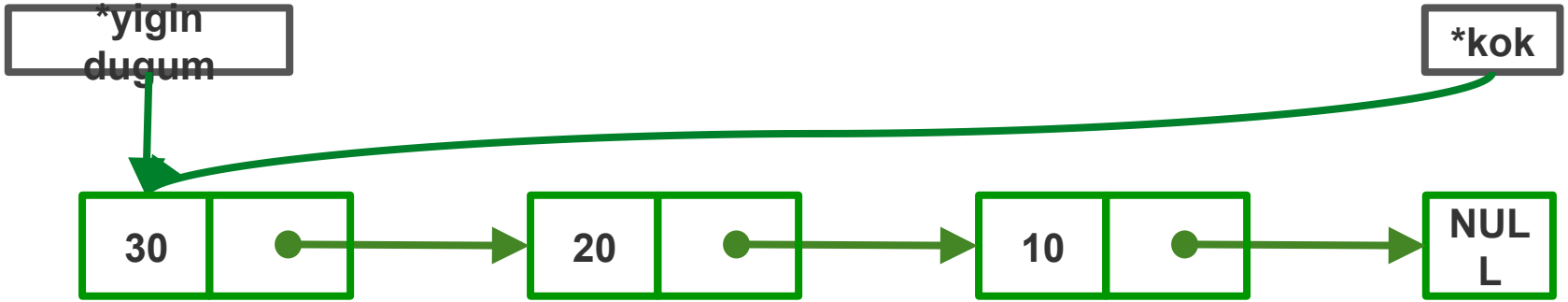
```
ekle(&kok, 10);
ekle(&kok, 20);
ekle(&kok, 30);
```



Yıgın(Baglı Liste) & C

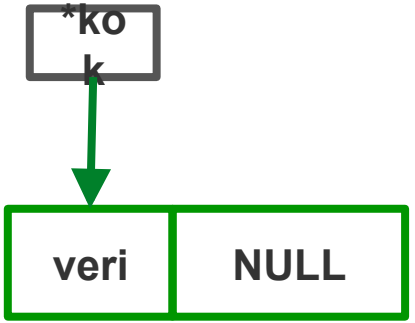
```
void ekle(struct YiginDugum** kok, int veri)
{
    struct YiginDugum* yigindugum = YeniDugum(veri);
    yigindugum->sonraki = *kok;
    *kok = yigindugum;
    printf("%d verisine sahip dugum eklendi\n", veri);
}
```

```
ekle(&kok, 10);

ekle(&kok, 20);

ekle(&kok, 30);
```



Yıgın(Baglı Liste) & C

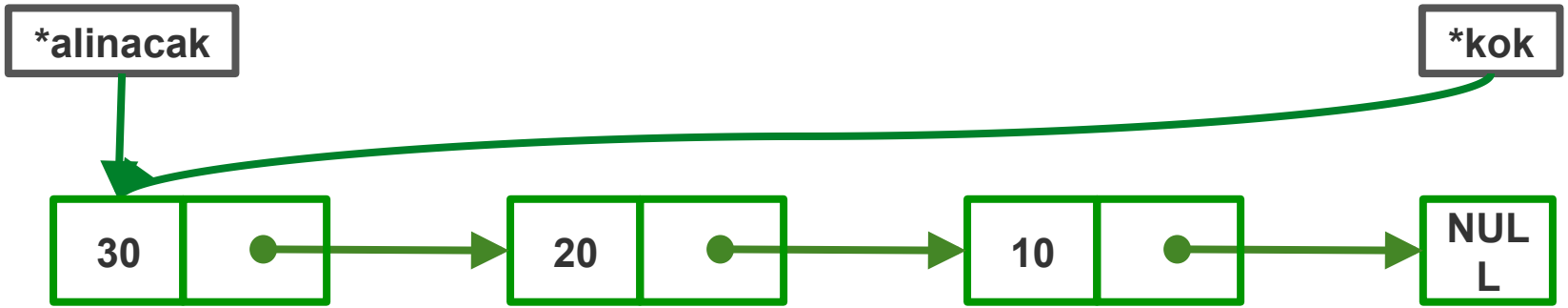
```
int bosmu(struct YiginDugum *kok)
{
    return !kok;
}
```



Yıgın(Baglı Liste) & C

```
int al(struct YiginDugum** kok)
{
    if (bosmu(*kok))
        return INT_MIN;
    struct YiginDugum* alinacak = *kok;
    *kok = (*kok)->sonraki;
    int alinan = alinacak->veri;
    free(alinacak);
    return alinan;
}
```

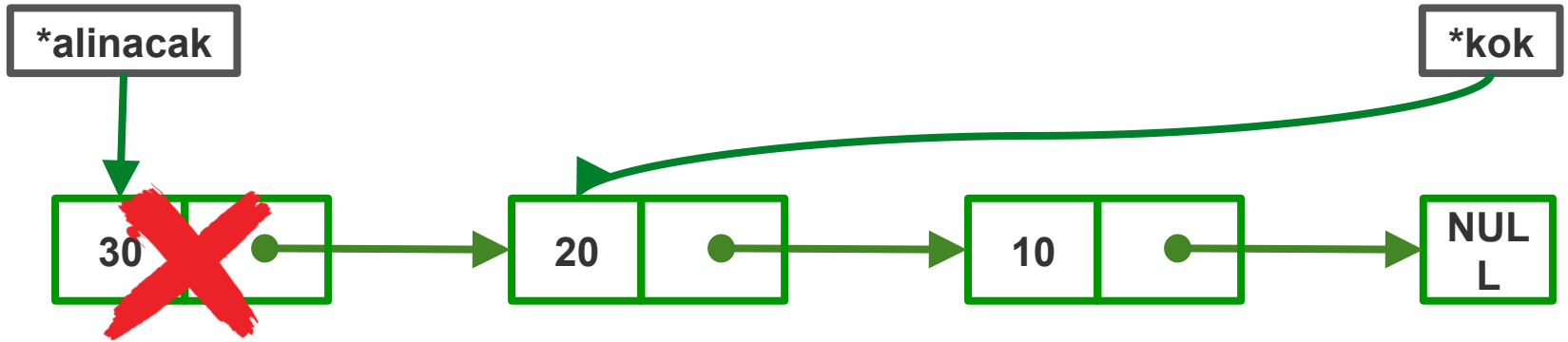
```
al(&kok)
```



Yıgın(Baglı Liste) & C

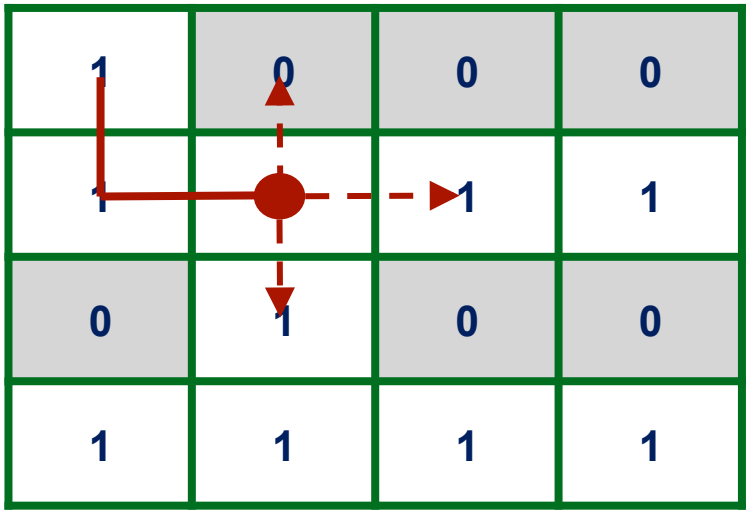
```
int al(struct YiginDugum** kok)
{
    if (bosmu(*kok))
        return INT_MIN;
    struct YiginDugum* alinacak = *kok;
    *kok = (*kok)->sonraki;
    int alinan = alinacak->veri;
    free(alinacak);
    return alinan;
}
```

```
al(&kok)
```

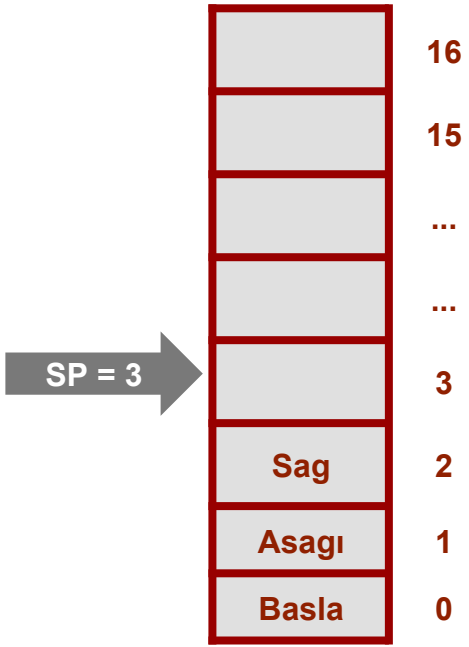


Labirent

Fare

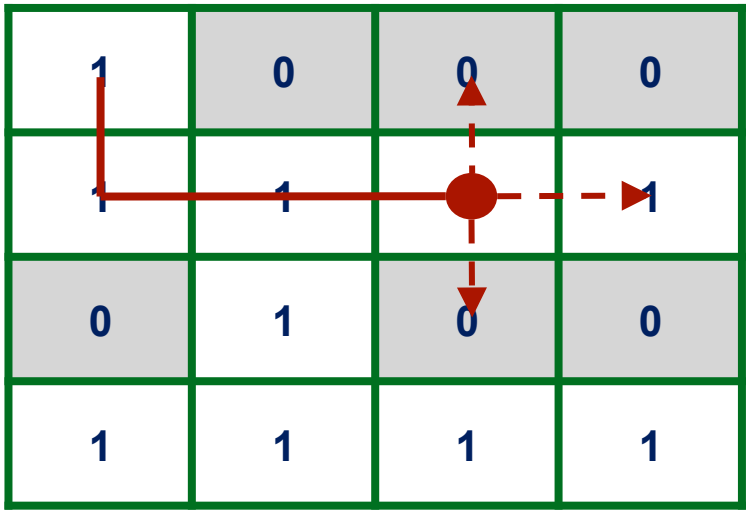


Peynir

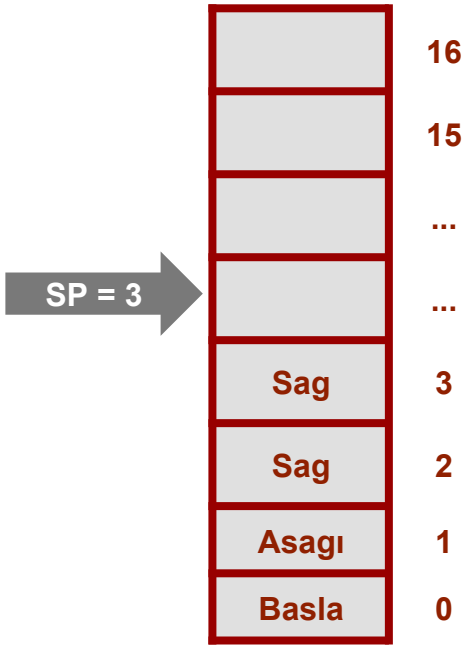


Labirent

Fare

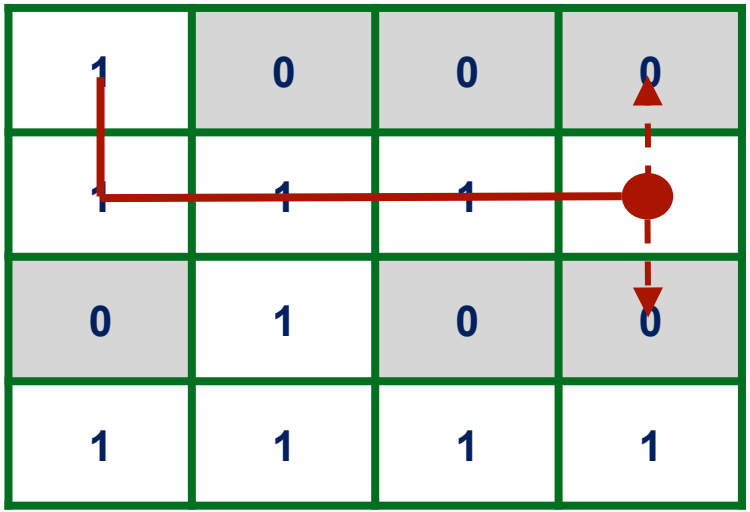


Peynir

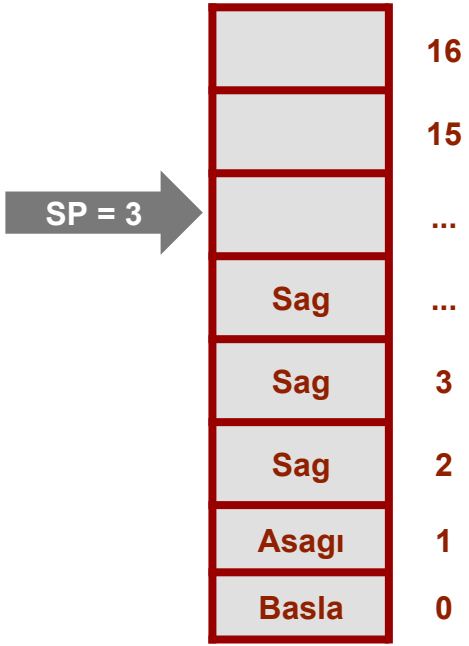


Labirent

Fare

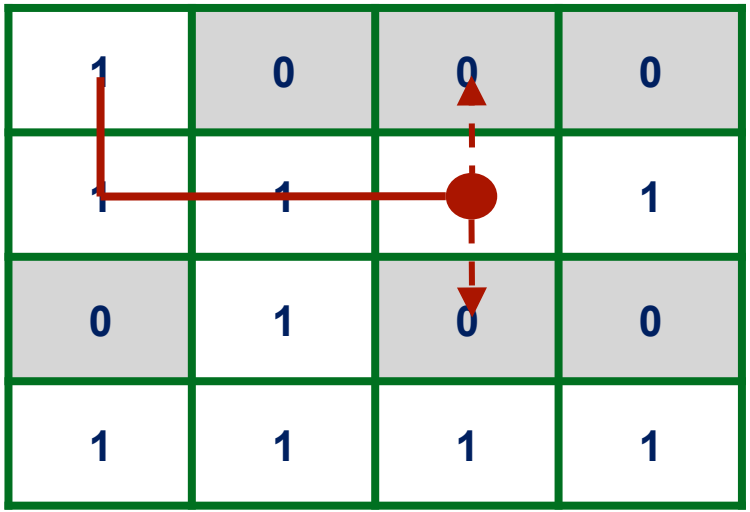


Peynir

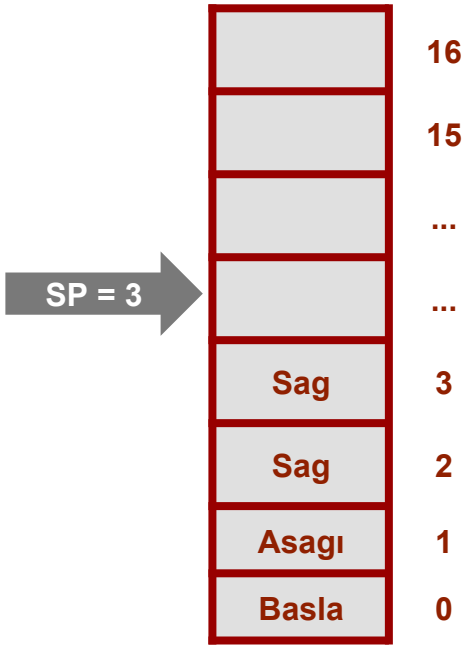


Labirent

Fare

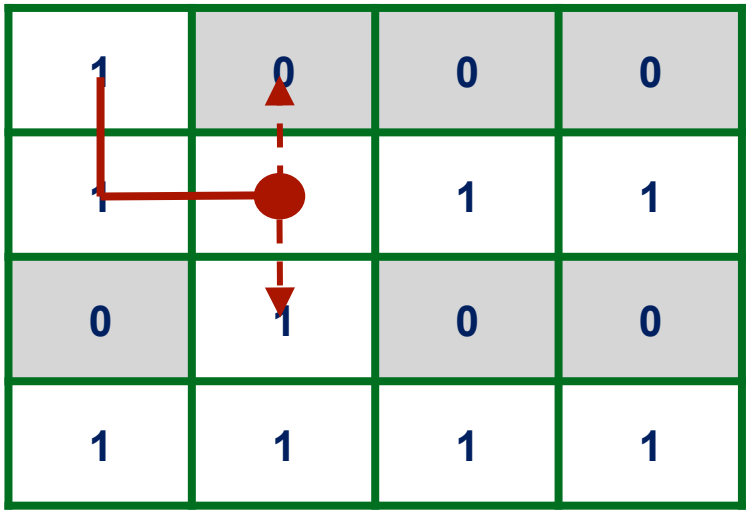


Peynir

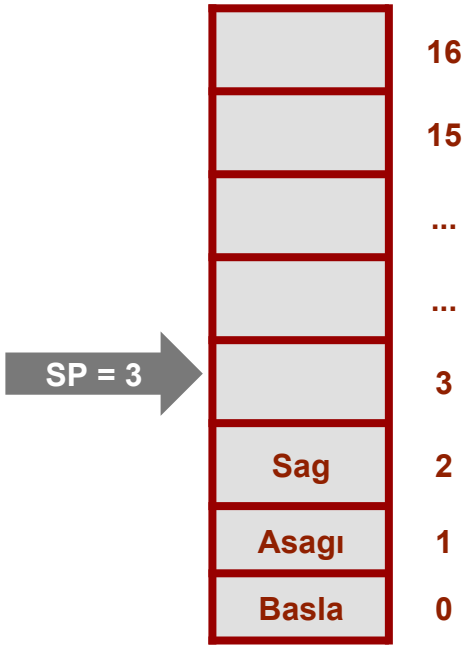


Labirent

Fare

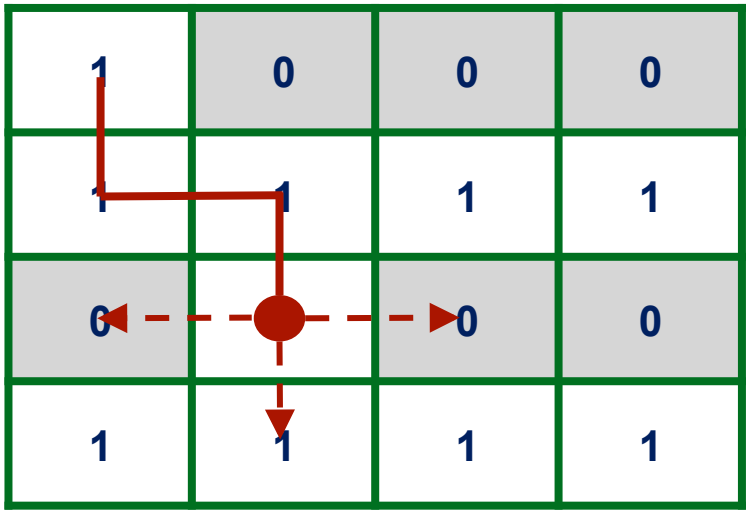


Peynir

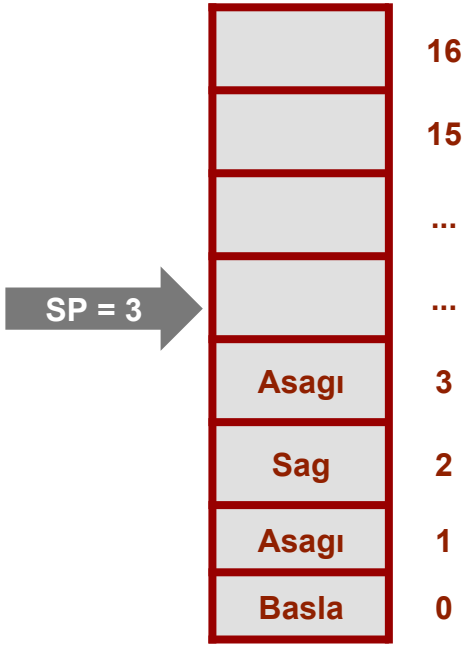


Labirent

Fare

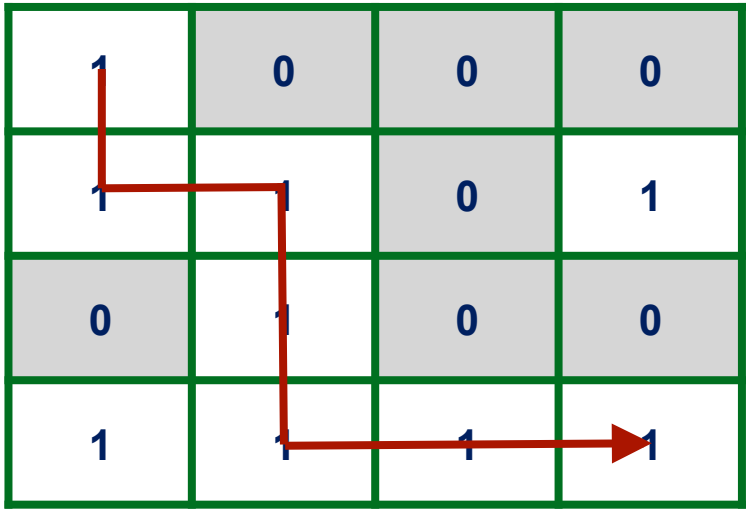


Peynir



Labirent

Fare



Peynir

Kuyruk

Her iki ucu açık
Bir uçtan veri (enqueue) ekleme
Diger uçtan veri (dequeue) çıkarma
First-In-First-Out

Gösterim


In
Data
Data
Data
Data
Data
Data
Out
Last In Last Out
First In First Out
Queue


LAST IN
LAST OUT
ONE WAY
FIRST IN
FIRST OUT
FIRST IN
FIRST OUT

Kuyruk Operasyonları

enqueue(): Sona elemanı ekler

dequeue(): En öndeki elemanı çıkartır

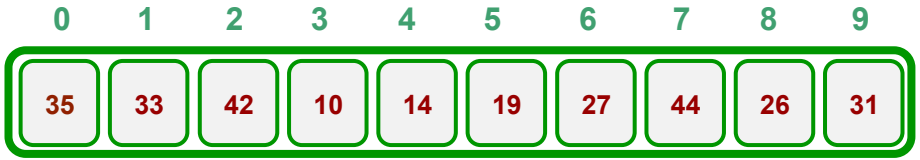
isfull(): Kuyruk dolu mu?

isempty(): Kuyruk bos mu?

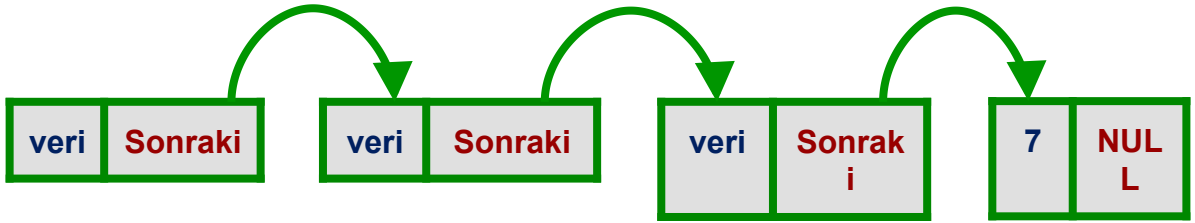
Kuyruk Gerçeklestirimi

Dizi

```
int
dizi[10]={35,33,42,10,14,19,27,44,26,31};
```

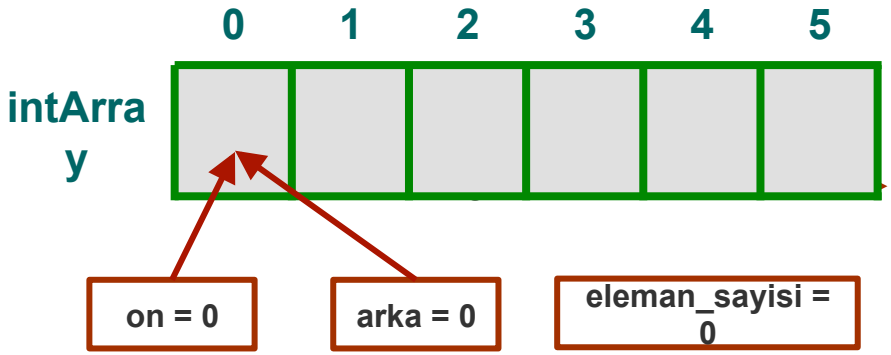


Baglantılı Listeler



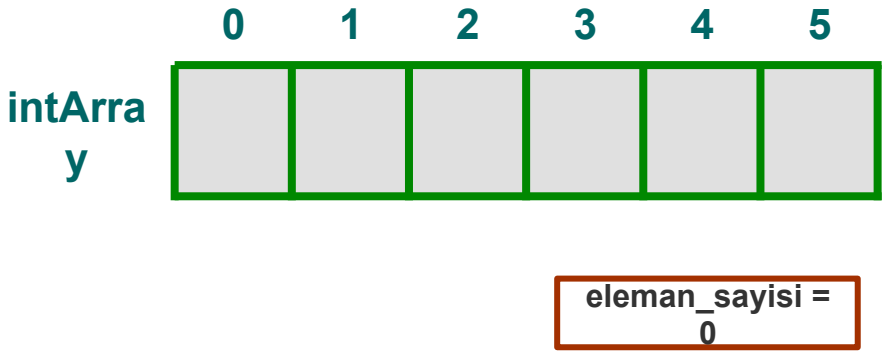
Kuyruk Gerçeklestirimi

```
#define kapasite 6

int intArray[kapasite];
int on = 0;
int arka = 0;
int eleman_sayisi=0;
```



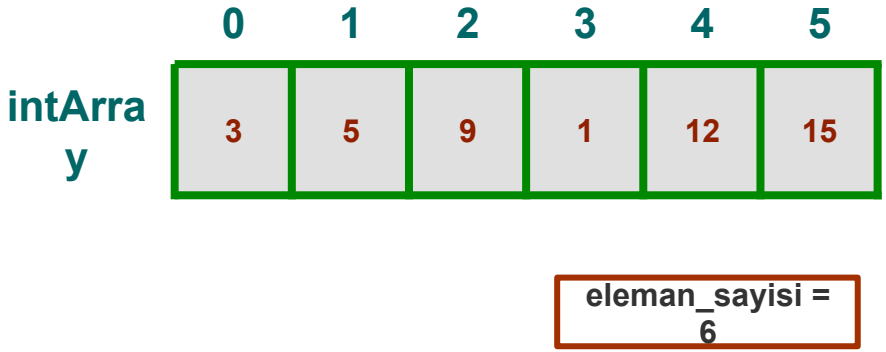
Kuyruk Gerçeklestirimi

```
bool bos_mu(){
    return eleman_sayisi==0;
}
```



Kuyruk Gerçeklestirimi

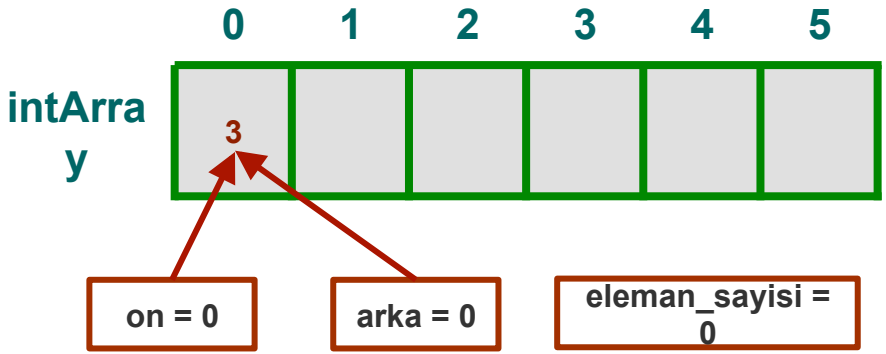
```
bool dolumu(){
    return eleman_sayisi==kapasite;
}
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

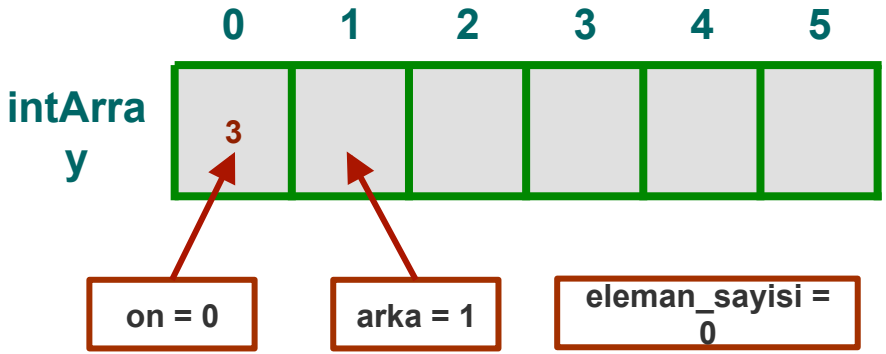
```
ekle(3);
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

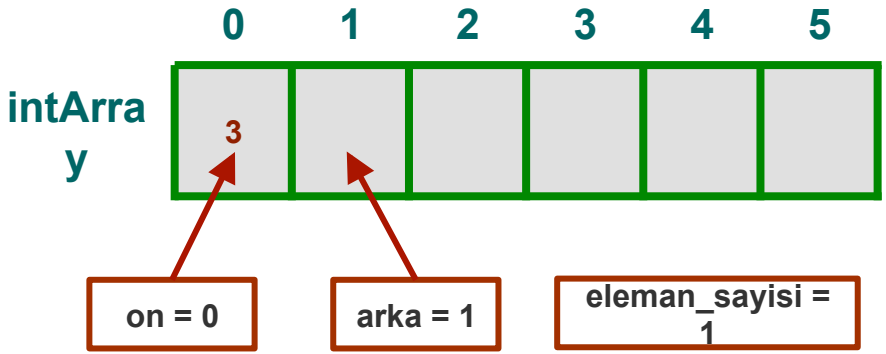
```
ekle(3);
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

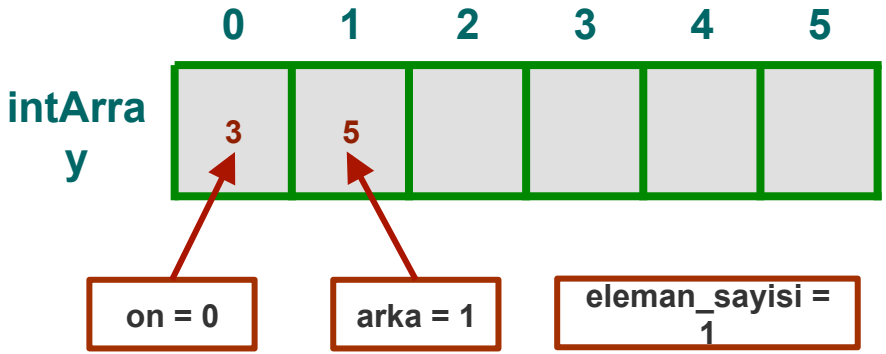
```
ekle(3);
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

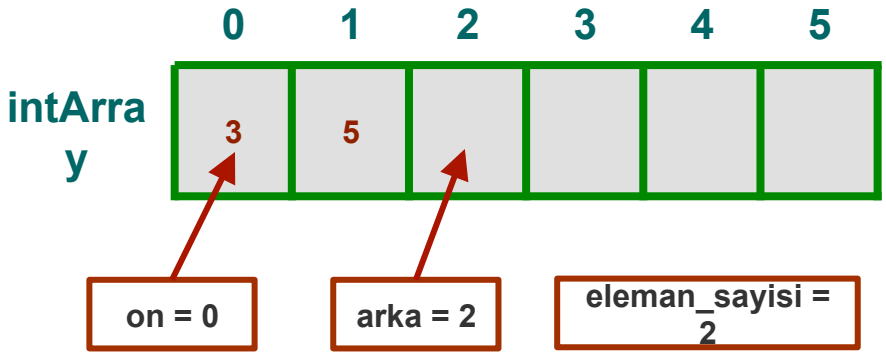
```
ekle(3);
ekle(5);
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

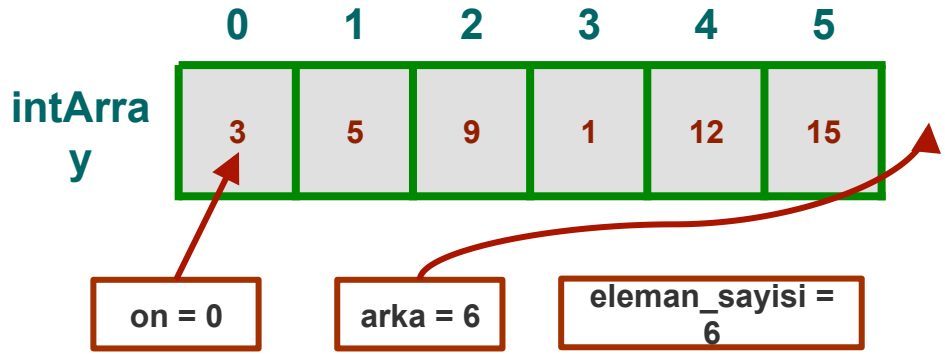
```
ekle(3);
ekle(5);
```



Kuyruk Gerçeklestirimi

```
int ekle(int veri){
    if(dolumu()) {
        printf("Kuyruk dolu!\n");
        return 0;
    }
    intArray[arka]= veri;
    arka++;
    if(arka==kapasite) arka=0;
    eleman_sayisi++;
    return 1;
}
```

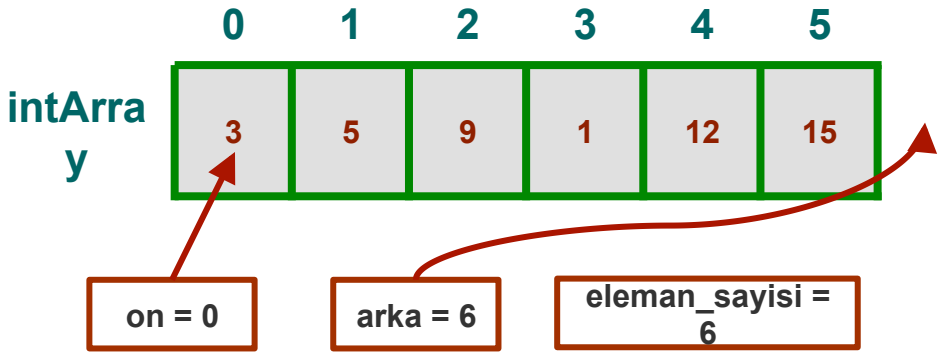
```
ekle(3);
ekle(5);
ekle(9);
ekle(1);
ekle(12);
ekle(15);
```



Kuyruk Gerçeklestirimi

```
int cikar(){
    if(bos_mu()){
        printf("Kuyruk boş!");
        return 0;
    }
    int data = intArray[on];
    on++;
    if(on==kapasite) on=0;
    eleman_sayisi--;
    return data;
}
```

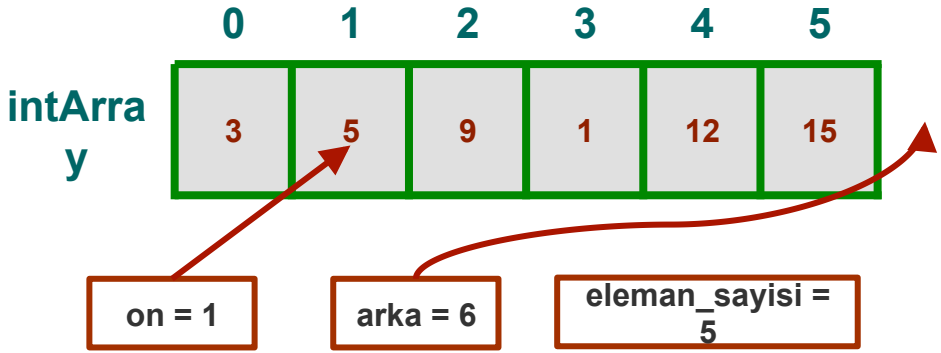
```
cikar( );
```



Kuyruk Gerçeklestirimi

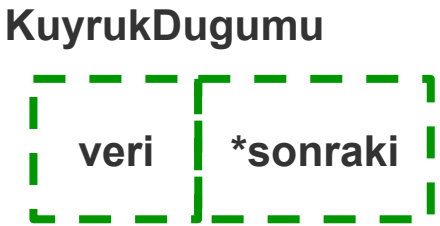
```
int cikar(){
    if(bos_mu()){
        printf("Kuyruk boş!");
        return 0;
    }
    int data = intArray[on];
    on++;
    if(on==kapasite) on=0;
    eleman_sayisi--;
    return data;
}
```

```
cikar( );
```



Kuyruk(Baglı Liste)

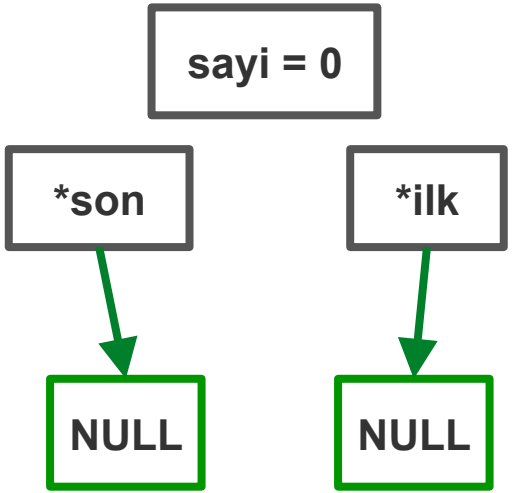
```
struct KuyrukDugumu {
        int veri;
        struct KuyrukDugumu *
sonraki;
};
```



Kuyruk(Baglı Liste)

```
class Kuyruk {
private:
        KuyrukDugumu * ilk;
        KuyrukDugumu * son;
        int _sayi;
public:
        Kuyruk() {
                this->_sayi = 0;
                this->ilk = this->son = NULL;
        }
        void ekle(int veri);
        int al();
        int sayi() { return this->_sayi; }
};
```

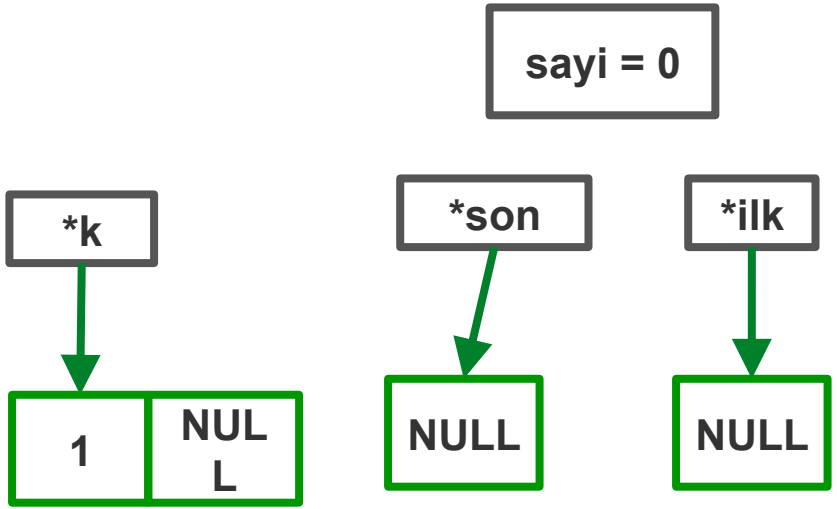
```
Kuyruk *k = new Kuyruk();
```



Kuyruk(Bagli Liste)

```
void Kuyruk::ekle(int veri) {
        // bagli listede sona ekleme islemi
        KuyrukDugumu *k = new KuyrukDugumu();
        k->veri = veri;
        k->sonraki = NULL;

        if (this->ilk == NULL) {
                this->ilk = this->son = k;
        } else {
                this->son->sonraki = k;
                this->son = k;
        }

        this->_sayi++;
}
```

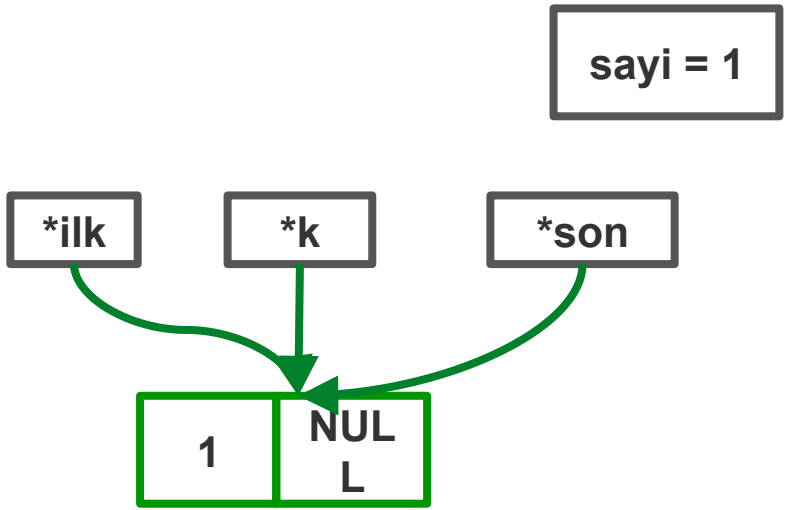
```
k->ekle(1);
```



Kuyruk(Bagli Liste)

```
void Kuyruk::ekle(int veri) {
        // bagli listede sona ekleme islemi
        KuyrukDugumu *k = new KuyrukDugumu();
        k->veri = veri;
        k->sonraki = NULL;

        if (this->ilk == NULL) {
                this->ilk = this->son = k;
        } else {
                this->son->sonraki = k;
                this->son = k;
        }

        this->_sayi++;
}
```

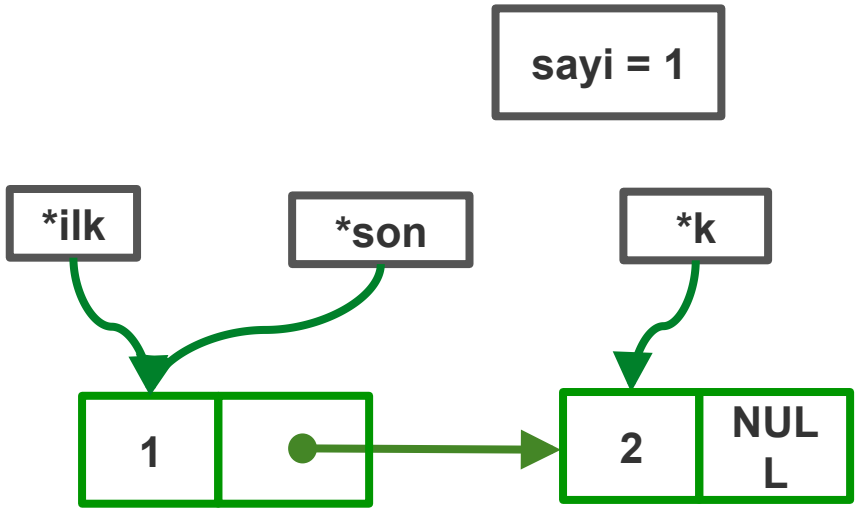
```
k->ekle(1);
```



Kuyruk(Bagli Liste)

```
void Kuyruk::ekle(int veri) {
        // bagli listede sona ekleme islemi
        KuyrukDugumu *k = new KuyrukDugumu();
        k->veri = veri;
        k->sonraki = NULL;

        if (this->ilk == NULL) {
                this->ilk = this->son = k;
        } else {
                this->son->sonraki = k;
                this->son = k;
        }

        this->_sayi++;
}
```

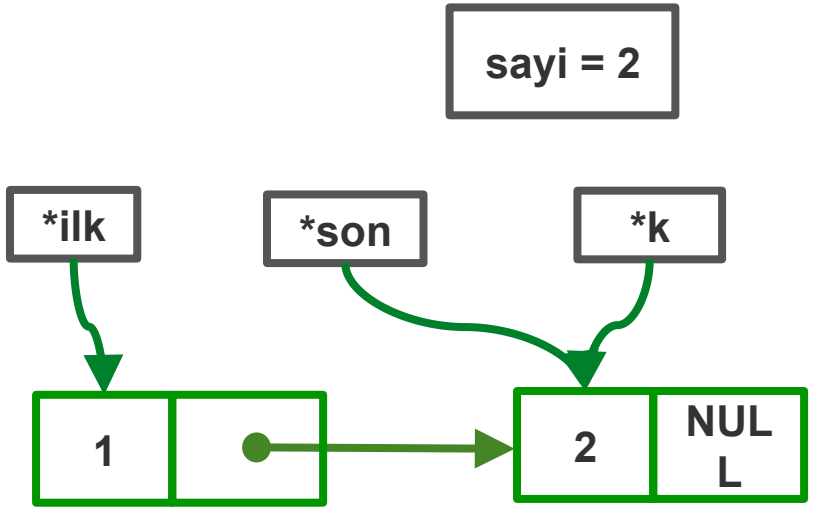
```
k->ekle(1);
k->ekle(2);
```



Kuyruk(Bagli Liste)

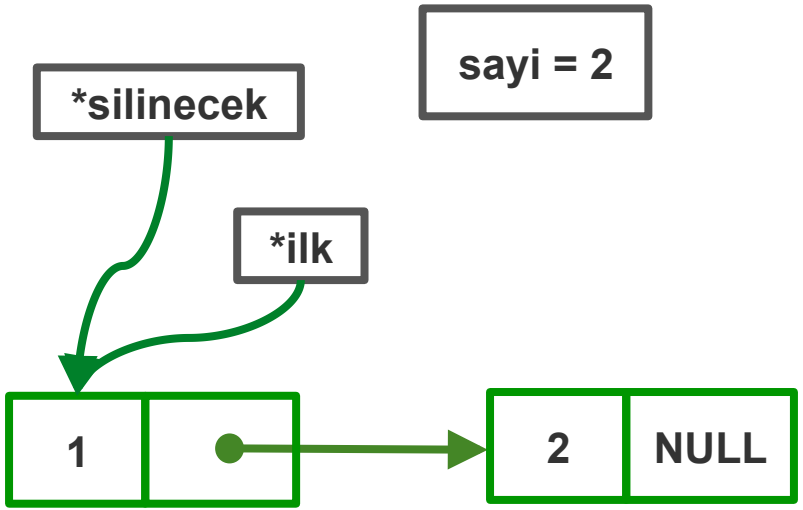
```
void Kuyruk::ekle(int veri) {
        // bagli listede sona ekleme islemi
        KuyrukDugumu *k = new KuyrukDugumu();
        k->veri = veri;
        k->sonraki = NULL;

        if (this->ilk == NULL) {
                this->ilk = this->son = k;
        } else {
                this->son->sonraki = k;
                this->son = k;
        }

        this->_sayi++;
}
```

```
k->ekle(1);
k->ekle(2);
```



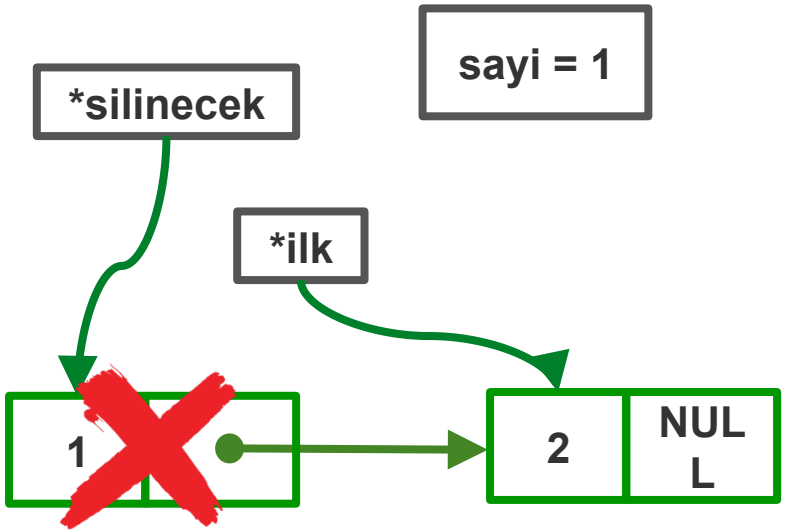
Kuyruk(Bagli Liste)

```
                                                                    k->al( );
int Kuyruk::al( ) {
        assert( this->sayi() != 0 ); // eleman yoksa hata ver
        int veri = this->ilk->veri;
        // bagli listede bastan silme islemi
        KuyrukDugumu * silinecek = this->ilk;
        this->ilk = this->ilk->sonraki;
        delete silinecek;

        this->_sayi--;
        return veri;
}
```



Kuyruk(Bagli Liste)

```
                                                                k->al( );
int Kuyruk::al( ) {
        assert( this->sayi() != 0 ); // eleman yoksa hata ver
        int veri = this->ilk->veri;
        // bagli listede bastan silme islemi
        KuyrukDugumu * silinecek = this->ilk;
        this->ilk = this->ilk->sonraki;
        delete silinecek;

        this->_sayi--;
        return veri;
}
```



Sorular