6. YIĞIN (STACK) VERİ YAPISI

Giriş

Yığın (*Stack*) veri yapısı basit olmasına karşılık güçlü bir veri yapısıdır ve günümüzde uygulamaların geliştirilmesi sırasında çok yaygın olarak kullanılır. Özellikle çözülmesi gereken problemin doğasından dolayı, elemanlara sadece bir noktadan sırayla erişilmesi gerektiği durumlarda yığın yapısı önemli bir araçtır. Örneğin, her web tarayıcısının bir *geri düğmesi* vardır. Web sayfaları arasında gezerken, bu sayfalar bir yığın üzerine yerleştirilir. Görüntülemekte olan mevcut sayfa en üstte ve bakılan ilk sayfa en alttadır. Geri düğmesine basılırsa, sayfalar arasında ters sırada hareket etmeye başlanır. Bu örnek yığın (Stack) veri yapısını tam olarak açıklar.

Bundan başka yığın veri yapısını masanın üzerinde üst üste duran logolara da benzetebiliriz. Şekil 6.1'de yığın veri yapısının daha iyi anlaşılabilmesi için üst üste duran logolar gösterilmiştir.

Şekil 6.1 Masada Üst Üste Logo Yığını

Üst üste duran bu logo yığınına bir logo daha eklemek istediğimizde, en üste bir logo yerleştirebiliriz veya yığından bir logo eksiltilmesini istediğimizde en üste duran logoyu alırız. Örneğin logo yığının en altındaki logoyu almak isteyelim; bunun için üsteki logoları yığından

sırayla birer çıkartıp en alttaki logoyu alabiliriz. Yukarıdaki açıklamalar yığınların çalışma prensibine örnek olarak gösterilebilir.

Yığın yapısına geçek hayattan başka örnekler de verebiliriz;

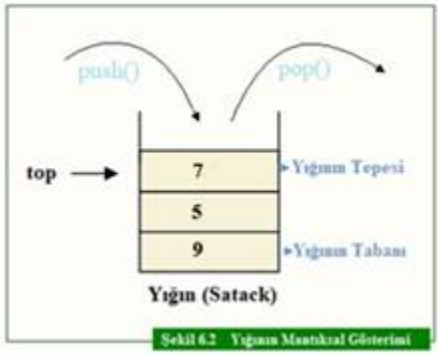
· Masa üzerinde üst üste duran kitap yığını

· Yemekhanelerde üst üste dizilmiş yemek tepsileri

· Mutfakta masa üzerinde üst üste duran tabaklar

Logo yığını örneğinde olduğu gibi, *eleman ekleme-çıkarma işlemlerinin sadece bir uçtan (en üstten - top) yapılabildiği veri yapılarına yığın (Stack) denir.* Başka bir ifade ile yığın, yeni öğelerin eklenmesi ve mevcut öğelerin kaldırılmasının, her zaman aynı uçtan gerçekleştiği sıralı bir öğe koleksiyonudur. Bazı kaynaklarda yığın veri yapısı *yığıt* olarak da isimlendirilmektedir.

Yığın veri yapısında ara elemanlara hiçbir zaman doğrudan erişilemez. Bu yapıda yığının elemanlarına son giren ilk çıkar (Last-In First-Out – LIFO) prensibi ile erişilir. Yığın veri yapısı soyut bir veri yapısıdır (Abstract Data Type –ADT). Kullanıcılar tarafından oluşturulur. Yığın veri yapısında elemanların eklendiği ve çıkarıldığı noktaya yığının tepesi, ilk elemanın eklendiği noktaya da yığının tabanı denir. Yığının tabanı önemlidir, çünkü yığında depolanan, tabana daha yakın olan öğeler yığında en uzun süre kalan öğeleri temsil eder. En son eklenen öğe, ilk kaldırılacak konumda olan öğedir. Yığın veri yapısında daha yeni öğeler üstte, eski öğeler ise alttadır. Şekil 6.2'de bir yığının mantıksal yapısı gösterilmiştir.



Şekil 6.2 Yığının Mantıksal Gösterimi

Şekil 6.2' de yığın veri yapısında elemanların yığına son giren ilk çıkar (LIFO) prensibine göre sıralı olarak eklenip çıkarıldığı açıkça görülmektedir.

Yığın veri yapısında, dizi veri yapısından farklı olarak veriye ancak belirli bir sırayla erişilebilir. Hatırlanacağı gibi dizi veri yapısında dizinin elemanlarına indis numarası ile doğrudan erişilebiliyordu. Dizi veri yapısı statik bir yapı (*Dizilerde eleman sayısı dizi bildirimi sırasınada yapılmakta, çalışma zamanı sırasında değiştirilememektedir*) olmasına karşılık yığın yapısı dinamiktir. Yani çalışma zamanı sırasında yığının eleman sayısı arttırılıp azaltılabilir (*Buna yığınların bağlı liste ile oluşturulması başlığı altında örnek verilecektir*).

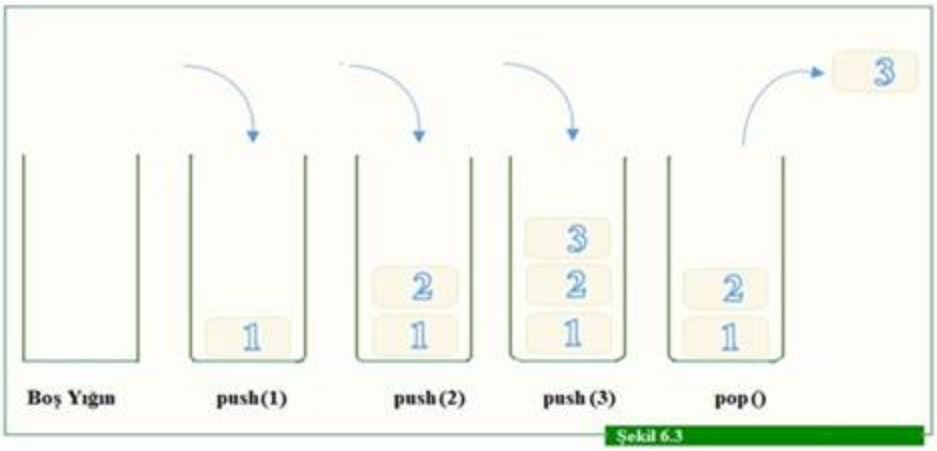
Yığın veri yapısı, bağlı liste veri yapısından, eleman ekleme-çıkarma işlemlerinin bir yığında sadece bir uçtan (en üstten - top) yapılması özelliği ile farklılaşmaktadır. Bilindiği gibi bağlı liste yapılarında eleman ekleme çıkarma işlemleri listenin başına, sonuna ve düğümlerin arasına yapılabiliyordu.

Yığınlar genellikle, programcılar tarafından geliştirilen programlarda sıklıkla kullanılsa da, program kullanırken kullanıcılar tarafından fark edilmez. Bunun nedeni, yığın oluşturma işlemlerinin bir uygulama çalışırken arka planda gerçekleştirilmesi ve kullanıcı tarafından görülmemesidir.

Programcıların, yığın oluşturulması sırasında konuyu doğru şekilde ele almaları önemlidir. Aksi halde yığın için yeterli belleğin kalmadığı durumlarda yığın taşması (*Stackoverflow*) oluşur ve bu programın çökmesine neden olabilir.

6.1. Yığın Üzerinde Yapılabilecek İşlemler

Şekil 6.3'ten anlaşılabileceği gibi yığın yapısında, ilki yığına eleman ekleme (*push*) ve ikincisi yığından eleman çıkarma (pop) olmak üzere iki temel işlem vardır. Bu işlemlerin dışında yığın işlemlerinde yaygın olarak kullanılan fonksiyonlar tablo 6.1'de verilmiştir.



Şekil 6.3

Fonksiyon	İşlev
push(int s)	Yığına eleman eklemek için kullanılır. Parametre alır.
pop()	Yığından eleman çıkarmak için kullanılır. Parametre almaz.
peek()	Yığının tepe elemanını döndürür. Parametre almaz.
empty()	Yığını boş/dolu kontrol etmek için kullanılır.
size()	Yığının eleman sayısını döndürür.

Tablo 6.1

6.2. Yığınların (Stack) Dizi (Array) Uygulaması

Yığın veri yapısı, tek boyutlu bir dizi kullanılarak gerçekleştirilebilir. Ancak dizi kullanılarak oluşturulan yığınlarda yalnızca sabit sayıda veri değeri depolanabileceği unutulmamalıdır. Çünkü daha önceki bölümlerde de açıklandığı gibi çalışma zamanı sırasında dizilerin boyutu değiştirilemez.

Yığınların dizi uygulaması oldukça basittir. Bu uygulama, önce tek boyutlu bir dizi tanımlanması ve daha sonra son giren - ilk çıkar *(LIFO)* prensibi kullanılarak, *"top"* adı verilen bir değişken yardımıyla değerlerin diziye eklenmesi veya silinmesi ilkesine dayanır. *"top"* adlı değişken her zaman yığının en üstteki öğesini gösterir, değeri yığının en üstteki elemanının indeksi ile işaret edilen dizi elemanının değerine eşittir.

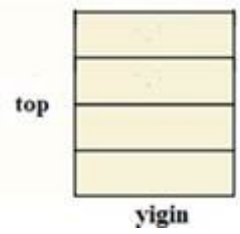
Yığınların dizi uygulamasında *"top"* değişkenine başlangıç değeri olarak -1 atanır ve daha sonra yığına bir değer eklemek istendiğinde, *"top"* değişkenin değeri bir arttırılır. Yığından bir eleman silinmesi istendiğinde, *"top"* değeri bir azaltılır.

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

#define BOYUT 4

 int yigin[BOYUT];
 int top=-1;
```

Yukarıda *#define* önişlemci komutu kullanılarak önce "BOYUT" tanımlanmıştır. Bu, program içerisinde BOYUT görülen her yere 4 (dört) yazılacağı anlamına gelir. İstendiğinde tanımdaki 4 (dört) değeri değiştirilerek dizinin boyut arttırılıp azaltılabilir. BOYUT tanımından sonra tamsayı tipinde bir *dizi* ve *top* değişkeni tanımlanmış ve *top* değişkenine -1 ilk değer olarak atanmıştır. Yukarıda verilen tanımlar yapıldıktan sonra aşağıdaki şekilde gösterilen durum (boş yığın) oluşur.



Oluşan bu son durumda, *top* değişkeninin herhangi bir elemana (henüz yığında herhangi bir eleman yok) ait indis değeri tutmadığına dikkate ediniz.

6.2.1. Yığınların Dizi Uygulamasında Yığına Eleman Ekleme (push(eklEleman))

```c
void push(int yigin[],int eklEleman){
    if(top==BOYUT-1)    {
        printf("\nYığın Dolu. Eleman Eklenemez\n");
        return;
    }
    top++;
    yigin[top]=eklEleman;
}
```

Program 6.1 Yığına Eleman Ekleme (Dizi uygulaması)

Program 6.1 'de verilen fonksiyon (*push(eklEleman*)) kullanılarak yığına eleman eklenebilir. 6.1'de verilen bu fonksiyondan görülebileceği gibi *top* değişkenin değeri bir arttırılarak bu indisin gösterdiği yere eleman ekleniyor. Ancak yığınların dizi uygulamasının önemli bir

sınırlılığı vardır. Dizini boyutu sabittir ve eleman eklem işlemi yapılmadan önce dizide boş yer olup olmadığı kontrol edilmelidir.

Eğer yığın bir dizi üzerinde oluşturuluyorsa bu yığına değer eklemek için aşağıdaki adımlar takip edilebilir:

1. **Adım:** Yığının dolu olup olmadığını kontrol et.

```
if(top==BOYUT-1)
```

2. **Adım:** Yığın dolu ise, kullanıcıya yığının dolu olduğu mesajını ver ve fonksiyondan çık.

```
printf("\nYığın Dolu. Eleman Eklenemez\n");
return;
```

3. **Adım:** Yığın dolu değil ise, top değişkeninin değerini bir arttır. Bu değerin gösterdiği yere elemanı ekle.

```
top++;
yigin[top]=eklEleman;
```

4. **Adım:** *push* fonksiyonu tekrar tekrar çağırılarak yığına dizinin eleman sayısı kadar eleman eklenebilir.

6.2.2. Yığınların Dizi Uygulamasında Yığından Eleman Silme (*pop()*)

```
void pop(int yigin[]){
    int silinecekEleman;
    if(top==-1){
        printf("Yığın Boş. Silinecek Eleman Yok\n");
        return;
    }
    silinecekEleman=yigin[top];
    top--;
    printf("%d Silme İşlemi Gerçekleşti\n",silinecekEleman);
    return;
}
```

Program 6.2 Yığından Eleman Silme (Dizi Uyg.)

Program 6.2 ‘de verilen fonksiyon (*pop()*) kullanılarak bir yığından eleman silinebilir. 6.2’de verilen *pop()* fonksiyondan görülebileceği gibi *top* değişkenin değeri bir eksiltilerek, indisi top olan eleman (yığının tepe elemanı) yığından çıkarılıyor. Silinen elemanın yığının tepesindeki eleman olduğuna dikkat ediniz.

Fonksiyonda silme işlemine başlamadan önce yığında eleman olup olmadığını kontrol etmek uygulamada beklenmeyen hataların önüne geçmek için önemlidir.

Eğer yığın bir dizi üzerinde oluşturulmuş ise, bu yığından değer silmek için aşağıdaki adımlar takip edilmelidir:

1. **Adım:** Yığının boş olup olmadığını kontrol et.

```
if(top==-1)
```

2. **Adım:** Yığın BOŞ ise, kullanıcıya uygun mesaj ver ve fonksiyonun çalışmasını sonlandır.

```
printf("Yığın Boş. Silinecek Eleman Yok\n");
return;
```

3. **Adım:** Yığın BOŞ DEĞİLSE, yığının tepesindeki elemanı *silinecekElaman* 'a ata ve *top* değerini 1 (bir) eksilt, silme işleminin gerçekleştirildiğine dair kullanıcıya mesaj ver ve fonksiyonun çalışmasını sonlandır.

```
silinecekEleman=yigin[top];
top--;
printf("%d Silme İşlemi Gerçekleşti\n",silinecekEleman);
return;
```

4. **Adım:** pop fonksiyon tekrar tekrar çağırılarak yığından eleman sime işlemine yığında eleman kalmayıncaya kadar devam edilebilir.

6.2.3. Dizi Uygulamasında Yığındaki Elemanları Yazdırma

```
void yazdir(int yigin[]){
    int i=0;
    if(top==-1){
        printf("Yığında Yadıracak Eleman Yok .\n");
        return;
    }
    printf("\n%d <-- top ",yigin[top]);
    for(i=top-1;i >=0;i--){
        printf("\n%d",yigin[i]);
    }
    printf("\n\n");
}
```

Program 6.3 Yığın Elemanlarını Yazdırma (Dizi Uyg.)

Program 6.3'te verilen fonksiyon bir yığının elemanlarını ekrana yazdırmak için geliştirilmiştir. Bir yığının elemanlarını yazdırmak için aşağıdaki adımları takip edebiliriz.

1. Adım: Yığının boş olup olmadığını kontrol et.

```
if(top==-1)
```

2. Adım: Yığın BOŞ ise, kullanıcıya uygun mesaj ver ve fonksiyonun çalışmasını sonlandır.

```
printf("Yığında Yadıracak Eleman Yok .\n");
return;
```

3. Adım: Yığın BOŞ DEĞİLSE, yığının *tepe* elemanını yazdırın.

```
printf("\n%d <-- top ",yigin[top]);
```

4. Adım: for döngüsü içerisinde top-1 'den başlayarak *top* değişkenin değeri 0 (sıfır) oluncaya kadar yığının elemanlarını ekrana yazdırın.

```
for(i=top-1;i >=0;i--){
    printf("\n%d",yigin[i]);
}
```

6.2.3. Yığınların Dizi Uygulaması (C programı)

Aşağıda Yığınların Dizi uygulamasına ait kodun tamamı verilmiştir.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define MAX 4
 int yigin[MAX],top;
/* Yığını Ekrana Yazdıran Fonksiy Deklarasyonu*/
void yazdir(int []);
/* Yığına Eleman Ekleyen Fonksiyon Deklarasyonu*/
void push(int [],int);
/* Yığını Eleman Silen Fonksiyon Deklarasyonu*/
void pop (int []);
/* Tepe Elemanı Yazdıran Fonksiyon Deklarasyonu*/
void peek(int []);
void main(){
    setlocale(LC_ALL,"Turkish");
    int eklEleman=0;
    int secim=0;
    top=-1;
    while(1){
        printf(" 1: Yığına Eleman Ekle (push)\n");
        printf(" 2: Yığından Eleman Sil\n");
        printf(" 3: Yığın Elemanlarını Yazdir\n");
        printf(" 4: Tepe Eleman\n");
        printf(" 5: Çıkış\n");
        printf(" Seçiminiz :");

scanf("%d",&secim);
switch(secim){
    case 1:
        printf("\nEklenecek Elemanı Giriniz :");
        scanf("%d",&eklEleman);
        printf("\n");
        push(yigin,eklEleman);
        break;
    case 2:
        pop(yigin);
        break;
    case 3:
        yazdir(yigin);
        break;
    case 4:
        peek(yigin);
        break;
    case 5:
        return 0;
    default:
    printf("\nGeçersiz Seçim\n\n");
    break; }}}
```

```
void yazdir(int yigin[]){
    int i=0;
    if(top==-1){
        printf("Yığında Eleman Yok .\n");
        return;
    }
    printf("\n%d <-- top ",yigin[top]);
    for(i=top-1;i >=0;i--){
        printf("\n%d",yigin[i]);
    }
    printf("\n\n");
}
void push(int yigin[],int eklEleman){
    if(top>=MAX-1 )    {
        printf("\nYığın Dolu. Eleman Eklenemez\n");
        return;
    }
    top++;
    yigin[top]=eklEleman;
}
void pop(int yigin[]){
    int silinecekEleman;
    if(top==-1){
        printf("Yığın Boş. Silinecek Eleman Yok\n");
        return;
    }
    silinecekEleman=yigin[top];
    top--;
    printf("%d Silme işlemi Başarılı\n",silinecekEleman);
    return;
}
void peek(int yigin[])
{
    printf("Yığının Tepe elemanı : %d \n\n",yigin[top]);
}
```

Program 6.4 Yığınların Dizi Uygulaması

6.3. Yığınların (Stack) Bağlı Liste (Linked List) Uygulaması

Yığınların dizi uygulaması, yanlızca sabit sayıda veri değeri ile çalışmaktadır. Eğer başlangıçta yığında kaç adet veri bulundurulacağı

bilinmiyorsa dizi kullanarak oluşturulan yığın veri yapısı uygun değildir. Bu problem, dizilerin çalışma zamanı sırasında boylarının azaltılıp arttırılamamsından kaynaklanmaktadır. Problemi aşmak için farklı bir uygulama yöntemi daha vardır. Bir yığın veri yapısı bağlı liste yapısı kullanılarak da oluşturulabilir. Bağlı liste yapısı kullanarak oluşturulan yığınlar sınırsız sayıda veri için çalıştırılabilir. Bunun için başlangıçta boyutu sabitlemeye gerek yoktur. Yığınların bağlı liste uygulamalarındaki bu özellik, yığın kullanarak oluşturulacak çözümlerde programcılara önemli esneklikler sağlar.

Yığınların bağlı liste uygulamalarında, yığına eklenen her yeni eleman, tepe eleman (top) olarak eklenir. Bu işlem tek yönlü bağlı listelerin başına eleman ekleme işlemi ile aynıdır. Aynı şekilde yığından eleman çıkarma işlemleri de tek yönlü bağlı listelerin başından eleman silme işlemine benzer. Bu nedenle yığınların bağlı liste uygulamaları kolay anlaşılabilir uygulamalardır.

```c
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
struct Dugum
{
   int veri;
   struct Dugum *sonraki;
}*top = NULL;
```

Yukarıda yığınların bağlı liste uygulamasına ait C programının başlangıç kısmındaki bazı satırlar verilmiştir. Verilen satırlardan da görülebileceği gibi, önce başlık dosyaları programa dahil edilmiş sonra da tek yönlü bağlı liste düğüm yapısı tanımlanmıştır. Daha sonra ‘*top’ işaretçisi tanımlanmış ve bu işaretçiye *NULL* değeri atanmıştır.

6.3.1. Yığınların Bağlı Liste Uygulamasında Yığına Eleman Ekleme (*push(eklEleman)*)

```
void push(int deger)
{
   struct Dugum *yeniDugum;
   yeniDugum = (struct Dugum*)malloc(sizeof(struct Dugum));
   yeniDugum->veri = deger;
   if(top == NULL)
      yeniDugum->sonraki = NULL;
   else
      yeniDugum->sonraki = top;
   top = yeniDugum;
   printf("\nEleman Eklendi!!!\n");
}
```

Program 6.5 Yığınların Bağlı Liste Uygulamasında Eleman Ekleme

Program 6.3'te görüldüğü gibi yığına eleman ekleyecek fonksiyon bir parametre almıştır (*push(**int deger**)).* Fonksiyonda önce bir düğüm oluşturulmuş ve bellekte bu düğüm için alan ayrılmıştır. Sonra parametre olarak fonksiyona gönderilen değer, oluşturulan düğüme atanmıştır (*yeniDugum ->veri = deger*). Yığına eleman eklemek için bu ifadeden sonraki işlemler aşağıda verilen sırayla gerçekleştirilmiştir.

1. Adım: Yığının boş olup olmadığını kontrol et

```
(top == NULL)
```

2. **Adım:** Yığın boş ise

```
yeniDugum->sonraki = NULL;
```

atamasını yap.

3. **Adım:** Yığın boş değilse:

3.1. Önceki '*top*' düğümünü *yeniDugum->sonraki* düğümüne atayın

```
yeniDugum->sonraki = top;
```

3.2. 'top' düğümünü yeniDugum olarak güncelleyin

```
top = yeniDugum;
```

Bu adımların sonunda yığına bir eleman eklenmiş olur. Yığına yeni eleman eklemek için fonksiyonu her defasında eklenmesi istenilen değerle çağırmak gerekecektir.

Yığınların dizi uygulamasında, yığına eleman eklenirken önce yığının dolu olup olmadığı kontrol ediliyor ve yığın dolu ise kullanıcıya bu doğrultuda mesaj verilip fonksiyondan çıkılıyordu. Yığınların bağlı liste uygulamasında ise bu şekilde bir kontrole ihtiyaç duyulmamıştır. Çünkü bu uygulamada önceden yığınına kaç eleman eklenceğini sınırlayan bir

dizi tanımlaması kullanılmamaktadır. Yığınların bağlı liste uygulaması ile oluşturulması sırasında yığına eklenecek eleman sayısını sınırlayan tek unsur bilgisayar belleğinin kapasitesidir. Bellekte yer olduğu sürece yığına eleman eklenebilir. Bu durum yığınların bağlı liste uygulamasının önemli bir avantajıdır.

6.3.2. Yığınların Bağlı Liste Uygulamasında Yığından Eleman Silme (*pop( )*)

```
void pop()
{
   if(top == NULL)
      printf("\nYığın Boş - Silinecek Eleman Yok!!!\n");
   else{
      struct Dugum *temp = top;
      printf("\nSilinen Eleman: %d", temp->veri);
      top = temp->sonraki;
      free(temp);
   }
}
```

Program 6.6 Yığınların Bağlı Liste Uygulamasında Eleman Silme

1. Adım: Yığının boş olup olmadığını kontrol edin

```
(top == NULL)
```

2. **Adım:** Yığın Boş ise kullanıcıya “Silinecek Eleman Yok” mesajı verip fonksiyondan çıkın.

3. **Adım:** Yığın Boş Değil ise:

3.1. Geçici bir düğüm oluşturun ve listenin ilk düğümünü (‘*top*’) oluşturulan bu düğüme atayın

```
struct Dugum *temp = top;
```

3.2. ‘top’ düğümünü temp->sonraki ile güncelleyin

```
top = temp->sonraki;
```

3.3. ‘temp’ düğümünü silin

```
free(temp);
```

6.3.3. Yığınların Bağlı Liste Uygulamasında Tepe Eleman (*peek( )*)

```
void peek()
{
   if(top == NULL)
      printf("\n Yığında Eleman Yok!!!\n");
   else{
      struct Dugum *temp = top;
      printf("\nTepe Eleman: %d", temp->veri);
         }
}
```

Yığın Boş ise Tepe Eleman Yoktur

Yığın Boş değilse 'temp' düğümü oluşturulup ilk düğüm temp'e atanır ve temp->veri yazdırılır

Program 6.7 Yığınların Bağlı Liste Uygulamasında Tepe Eleman

6.3.4. Yığınların Bağlı Liste Uygulamasında Yığın Boş/Dolu (*empty( )*)

```
void empty(){
    if(top == NULL)
      printf("\n Yığın Boştur!!!\n");

   else
    printf("\n Yığın Boş Değildir!!!\n");
}
```

(top==NULL) ise Yığın Boştur

(top==NULL) değilse yığın boş değildir.

Program 6.8 Yığın Boş / Yığın Boş Değil

6.3.5. Yığınların Bağlı Liste Uygulaması (C programı - toplu)

```
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
struct Dugum{
   int veri;
   struct Dugum *sonraki;
}*top = NULL;
void push(int);
void pop();
void yazdir();
void peek();
void empty();
}
void main(){
   int secim, deger;
   setlocale(LC_ALL,"Turkish");
printf("\n:Bağlı Liste Kullanarak Yığın\n");
   while(1){
      printf("\n****** MENU ******\n");
      printf("1. Eleman Ekle (push)\n");
      printf("2. Eleman Sil (pop)\n");
      printf("3. Tepe Eleman (peek)\n");
      printf("4. Boş/Dolu\n");
      printf("5. Yazdir\n");
      printf("6. Çıkış\n");
      printf("Seçiminiz  : ");
      scanf("%d",&secim);
      switch(secim){
      case 1:
       printf("Eklemek istediginiz Eleman:");
       scanf("%d", &deger);
       push(deger);
          break;
      case 2: pop(); break;
      case 3: peek(); break;
      case 4: empty(); break;
      case 5: yazdir();break;
      case 6: return 0;
      default: printf("\nHatalı Seçim!!!\n");
      }}}
```

```
void push(int deger){
   struct Dugum *yeniDugum;
   yeniDugum = (struct Dugum*)malloc(sizeof(struct Dugum));
   yeniDugum->veri = deger;
   if(top == NULL)
      yeniDugum->sonraki = NULL;
   else
      yeniDugum->sonraki = top;
   top = yeniDugum;
   printf("\nEleman Eklendi!!!\n");
}
void pop(){
   if(top == NULL)
      printf("\nYığın Boş - Silinecek Eleman Yok!!!\n");
   else{
      struct Dugum *temp = top;
      printf("\nSilinen Eleman: %d", temp->veri);
      top = temp->sonraki;
      free(temp);
   }}
void peek(){
   if(top == NULL)
      printf("\n Yığında Eleman Yok!!!\n");
   else{
      struct Dugum *temp = top;
      printf("\nTepe Eleman: %d", temp->veri);
         }}
void empty(){
    if(top == NULL)
      printf("\n Yığın Boştur!!!\n");

   else
    printf("\n Yığın Boş Değildir!!!\n");
}
void yazdir(){
   if(top == NULL)
      printf("\nYığın Boş - Yazdırılacak Eleman Yok!!!\n");
   else{
      struct Dugum *temp = top;
      while(temp->sonraki != NULL){
     printf("%d--->",temp->veri);
     temp = temp -> sonraki;
      }
      printf("%d--->NULL",temp->veri);
   }}
```

Yığınların Bağlı Liste Uygulaması (C Programı - Toplu)

6.4. Yığınların (Stack) Kullanım Yerleri

Yığın veri yapısı, giriş bölümünde de vurgulandığı gibi bilişim alanında birçok uygulamada kullanılmaktadır. Burada yığın veri yapısının en çok bilenen uygulamalarından bazıları sunulacaktır.

1. Web tarayıcılardaki *geri* (*back*) butonu

belki de yığınlara en sık verilen örnektir. Internet kullanan herkesin bildiği gibi, tarayıcıların geri butonu, en son ziyaret edilen sayfaya ilk önce dönülmesini sağlar. En son ziyaret edilen sayfadan önceki sayfaya dönülmek istendiğinde geri butonuna bir kere daha basmak gerekir. Bu tam bir yığın (stack) uygulamasıdır.

2. Ofis ve benzeri yazılım uygulamalarında bulunan *geri al (Undo)* butonunun

çalışma prensibi de web tarayıcılardaki geri (back) butonuna benzemektedir. Bu uygulamalardaki geri al butonu en son yapılan işlemin geri alınmasını sağlar. En son yapılan işlemin geri alınması yığın kullanımı için açıklayıcı bir örnektir.

3. Uygulamalarda yer parantezlerin kontrolünde de yığın uygulaması kullanılmaktadır. Bilindiği gibi bir ifadede en son açılan parantez en önce kapatılmalıdır. Bu kural yığın tanımına uygundur.

4. Programlarda yer alan matematiksel ifadelerden beklenen sonuçların ele edilmesi işlem önceliklerinin doğru çalışmasına bağlıdır. Bu ifadelerde yer alan operatörlerin işlem önceliklerinin dikkate alınarak değerlendirilmesi yığın veri yapısı ile gerçekleştirilir.

Yığın veri yapısının kullanıldığı yerlere verilecek örnek sayısının yukarıda verilenlerden çok daha fazla olduğu hatırdan çıkarılmamalıdır.

Bölüm Özeti

Yığın (stack) veri yapısını öğrenecek ve yığın veri yapısını bağlı liste veri yapısı ve dizi veri yapısı ile karşılaştırabilecek,

Elaman ekleme-çıkarma işlemlerinin sadece bir uçtan (en üstten - top) yapılabildiği veri yapılarına yığın (Stack) denir. Başka bir ifade ile yığın, yeni öğelerin eklenmesi ve mevcut öğelerin kaldırılmasının her zaman aynı uçtan gerçekleştiği sıralı bir öğe koleksiyonudur. Yığın veri yapısında ara elemanlara hiçbir zaman doğrudan erişilemez. Bu yapıda yığının elemanlarına son giren ilk çıkar (Last - In First - Out – LIFO) prensibi ile erişilir. Yığın veri yapısı soyut bir veri yapısıdır (Abstract Data Type –ADT). Kullanıcılar tarafından oluşturulur.

Yığın veri yapısında, dizi veri yapısından farklı olarak veriye ancak belirli bir sırayla erişilebilir. Hatırlanacağı gibi dizi veri yapısında dizinin elemanlarına indis numarası ile doğrudan erişilebiliyordu. Yığın veri yapısı, bağlı liste veri yapısından da, eleman ekleme-çıkarma işlemlerinin bir yığında sadece bir uçtan (en üstten - top) yapılması özelliği ile farklılaşmaktadır. Yığın veri yapısı farklılaşan bu özellikleri ile birçok uygulamada kullanılmaktadır.

Yığına eleman ekleyebilecek, Yığından eleman çıkartabilecek,

Yığınlara eklemek için bu kitapta iki farklı yöntem ele alındı; yöntemlerden birincisi yığınların dizi uygulaması ikincisi ise yığınların bağlı liste uygulaması. Yığınların dizi uygulamasında yığına eleman eklenirken önce dizinin dolu olup olmadığı kontrol edildi ve dolu ise eleman eklem işlemi yapılmadan fonksiyondan çıkıldı. Eğer yığın dolu değilse ‘top’ değeri bir arttırılarak bu indisin işaret ettiği yere yeni eleman eklendi. Yığınların bağlı liste uygulamasında yığına eleman eklerken ise yığının dolu olup olmadığı kontrolüne gerek duyulmadı. Çünkü bu uygulamada dizi uygulamasından farklı olarak eleman eklemek için bir sınır yoktur (*Burada sınırın belleğin kapasitesi olduğunu belirtmemiz gerekiyor*). Bağlı liste uygulamasında yığına eleman ekleme işlemi tek yönlü bağlı listelerin başına eleman ekleme işlemine benzemektedir. Aynı şekilde yığınların bağlı liste uygulamasında yığından eleman çıkarma işlemi tek yönlü bağlı listenin başından eleman silme işlemine benzemektedir.

Bunlara ilave olarak bu bölümde yığının tepe elemanının tespitini yapacak fonksiyonun (*peek()*) ve yığının boş mu dolu mu olduğunu

tespitte kullanılacak fonksiyonun (*empty()*) nasıl oluşturulacağını öğrendik.

Bundan başka son olarak gündelik hayatta yaygın olarak kullanılan web tarayıcılar, ofis ve benzeri uygulamalar üzerinden yığın kullanımına örnekler verilerek konunun pekiştirilmesi amaçlandı.