

Stack (Yığın)

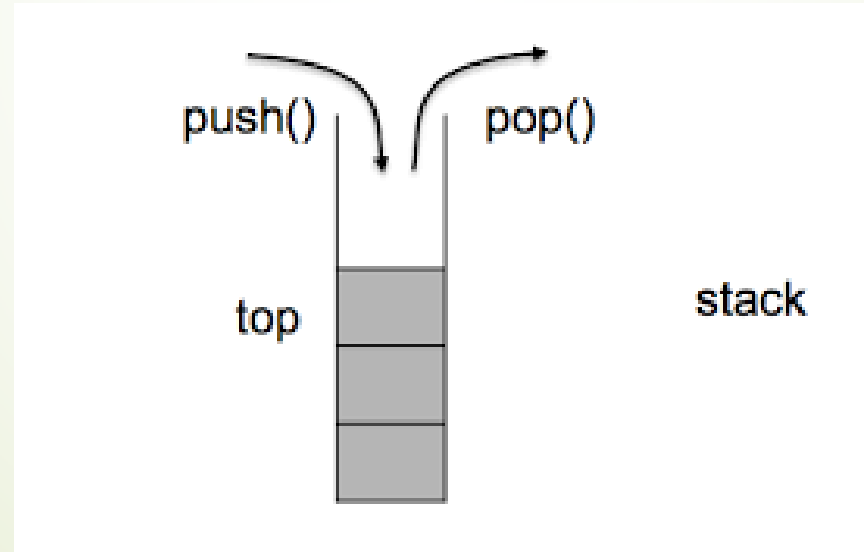
Eleman ekleme çıkarmaların en üstten (top) yapıldığı veri yapısına yığın (stack) adı verilir. Tek taraflı giriş ve çıkışlara açık olan. İlk giren son çıkar LIFO (Last in First Out) mantığı ile çalışan bir veri yapısı örneğidir.

Temelde iki veya üç fonksiyonu bulunur bunlar:

Push -> Stack içerisine bir bilgi koymaya (Stack'in en tepesine koyar)

Pop -> Stack içerisinden bir bilgi almaya (Stack'in en tepesinden alır)

Top -> Stack'in en tepesindeki bilgiyi alır ancak stackten çıkartmaz sadece okur



Stack (Yığın)

Bir yığın veya **FILO (ilk giren son çıkar)**, iki temel işlemle bir öge koleksiyonu görevi gören soyut bir veri türüdür: koleksiyona bir öge ekleyen *push* ve eklenen son ögeyi kaldıran *pop* 'tan oluşan mekanizmadır. Yığın halinde hem *push* hem de *pop* işlemi, yığının en üstünde bulunan aynı uçta gerçekleşir. Stack, doğrusal artan bir veri yapısı olup; *insert* (*push*) ve *delete* (*pop*) işlemleri, listenin sadece “top” adı verilen bir ucunda yani stack'in en üstünden gerçekleştirilir.

Ekleme: $O(1)$

Silme: $O(1)$

Erişim Süresi: $O(n)$ [En Kötü Durum]

Ekleme ve Silme işlemine bir uçta izin verilir.

Örnek: Yığınlar, işlev çağrılarını sürdürmek için kullanılır (en son çağrılan işlev önce yürütmeyi bitirmelidir), özyinelemeyi her zaman yığınların yardımıyla kaldırabiliriz. Yığınlar ayrıca, bir sözcüğü ters çevirmemiz, dengeli parantez olup olmadığının kontrol edilmesi ve en son yazdığınız kelimenin geri alma işlemini kullandığınızda ilk çıkardığınız editörlerde de kullanılır. Benzer şekilde, web tarayıcılarında geri işlevsellik uygulamak için.

Stack (Yığın)

Yığın İşlemleri

- **Ana yığın işlemleri:**

- ✓ *push(nesne)*: yeni bir nesne ekler

Girdi: Nesne Çıktı: Yok

- ✓ *pop()*: en son eklenen nesneyi çıkarıp geri döndürür.

Girdi: Yok Çıktı: Nesne

- **Yardımcı yığın işlemleri:**

- ✓ *top()*: en son eklenen nesneyi çıkarmadan geri döndürür.

Girdi: Yok Çıktı: Nesne

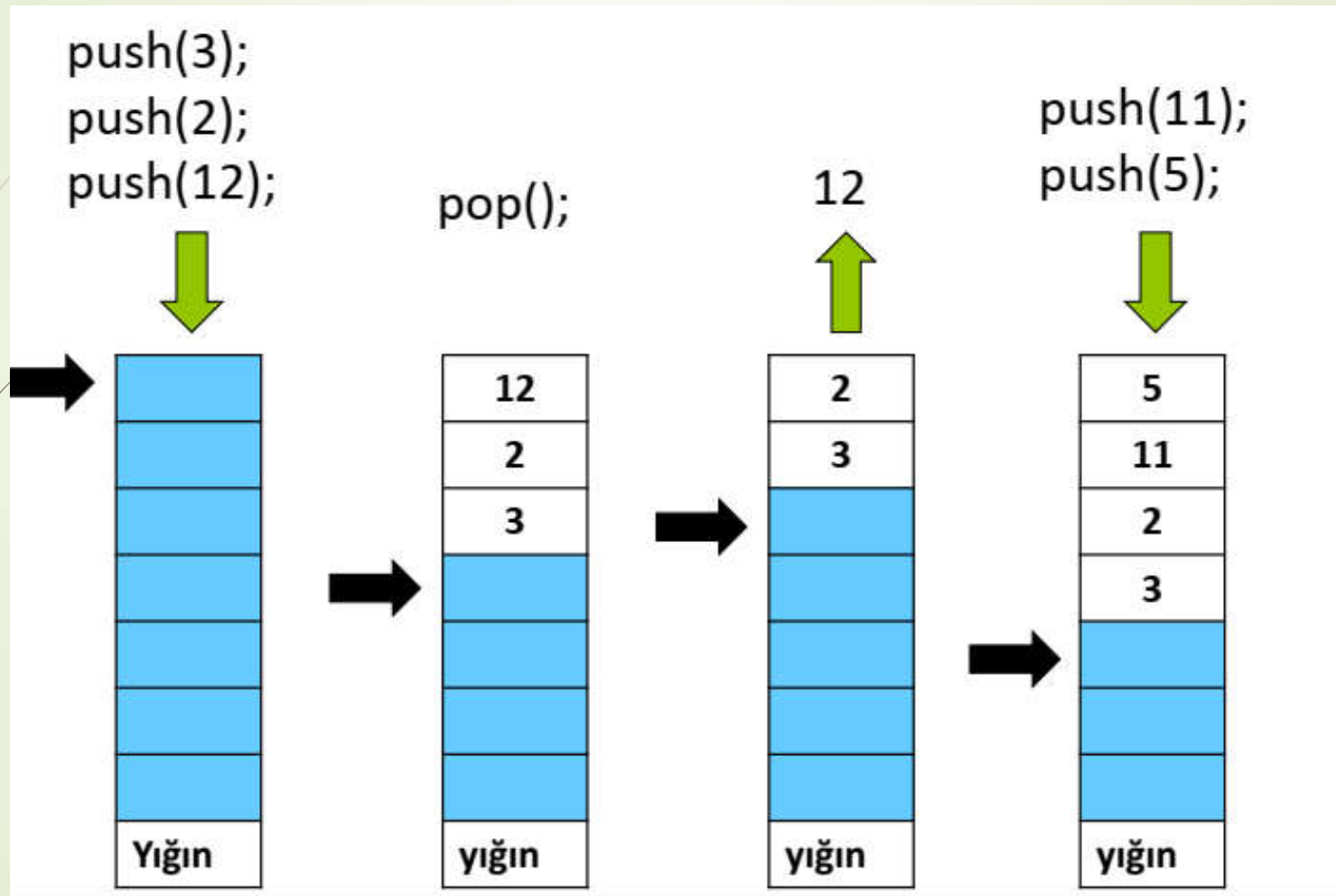
- ✓ *size()*: depolanan nesne sayısını geri döndürür.

Girdi: Yok Çıktı: Tamsayı

- ✓ *isEmpty()*: yığında nesne bulunup bulunmadığı bilgisi geri döner.

Girdi: Yok Çıktı: Boolean

Stack (Yığın)



Stack (Yığın)

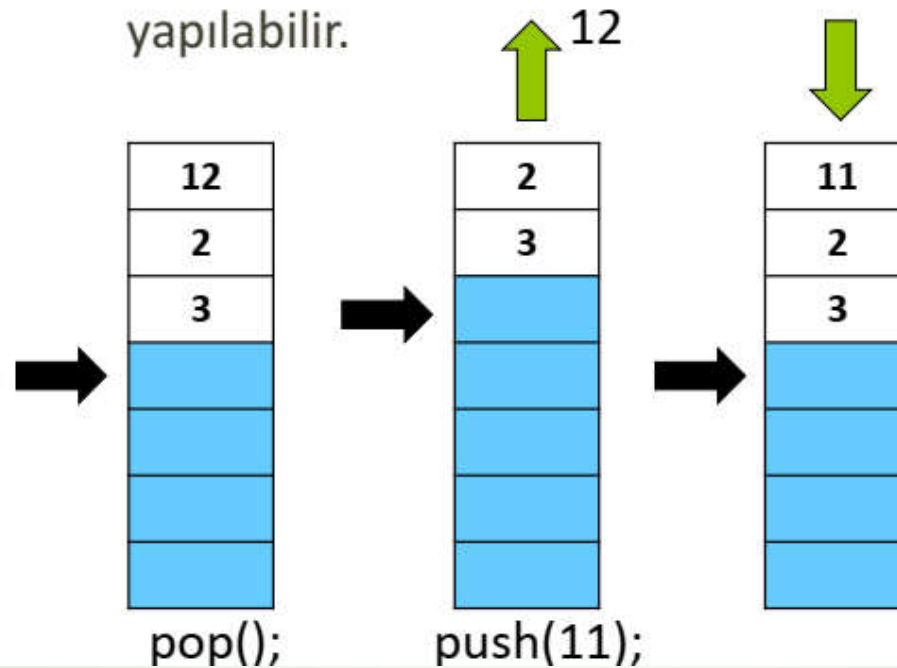
Örnek : Yığına ekleme ve çıkarma

İşlem	Yığın (tepe)	Çıktı
push("M");	M	
push("A");	MA	
push("L");	MAL	
push("T");	MALT	
pop();	MAL	T
push("E");	MALE	T
pop();	MAL	TE
push("P");	MALP	TE
pop();	MAL	TEP
push("E");	MALE	TEP
pop();	MAL	TEPE

Stack (Yığın)

Dizi Tabanlı Yığın

- Bir yığının gerçekleştirilmesinin en kolay yolu dizi kullanmaktır. Yığın yapısı dizi üzerinde en fazla N tane eleman tutacak şekilde yapılabilir.



Stack (Yığın)

```
class Yigin
{
    static int kapasite = 100;
    int[] yigit = new int[kapasite];
    int sayac = 0;

    //int itele(deger)
    //int cek()
    //bool bosmu()
    //bool dolumu()
    //int tepe()

    //EMPTY
    public bool bosmu()
    {
        if (sayac == 0)
        {
            Console.WriteLine("Stack boş!");
            return true;
        }
        else
            return false;
    }

    //FULL
    public bool doluMu()
    {
        if (sayac == kapasite - 1)
            return true;
        else
            return false;
    }
}
```

Stack (Yığın)

//PUSH

```
public int itele(int deger)
{
    if (doluMu())
        return -1;

    yigit[sayac] = deger;
    sayac++;
    return 1;
}
```

//POP

```
public int cek()
{
    if (bosmu())
    {
        Console.WriteLine("stack boş!");
        return -1;
    }
    else
    {
        return yigit[sayac--];
    }
}
```


Stack (Yığın)

```
//TOP
public int tepe()
{
    if (bosmu())
        return -1;
    else
        return yigit[sayac - 1];
}


static void Main(string[] args)
{
    Yigin y = new Yigin();
    y.itele(10);
    y.itele(5);
    y.cek();
    y.itele(8);
    Console.WriteLine("Stack top:"+y.tepe());
    y.cek();
    y.cek();
    y.cek();
    y.yazdir();
}

//PRINT
public void yazdir()
{
    for (int i = 0; i < sayac+1; i++)
    {
        Console.Write(yigit[i] + "-");
    }
}
```


Stack (Yığın)

Infix, Postfix, Prefix:

Infix

$((A / (B \wedge C)) - ((D * E) - (A * C)))$
 $((A / (B \wedge C)) - ((D * E) - (A * C)))$


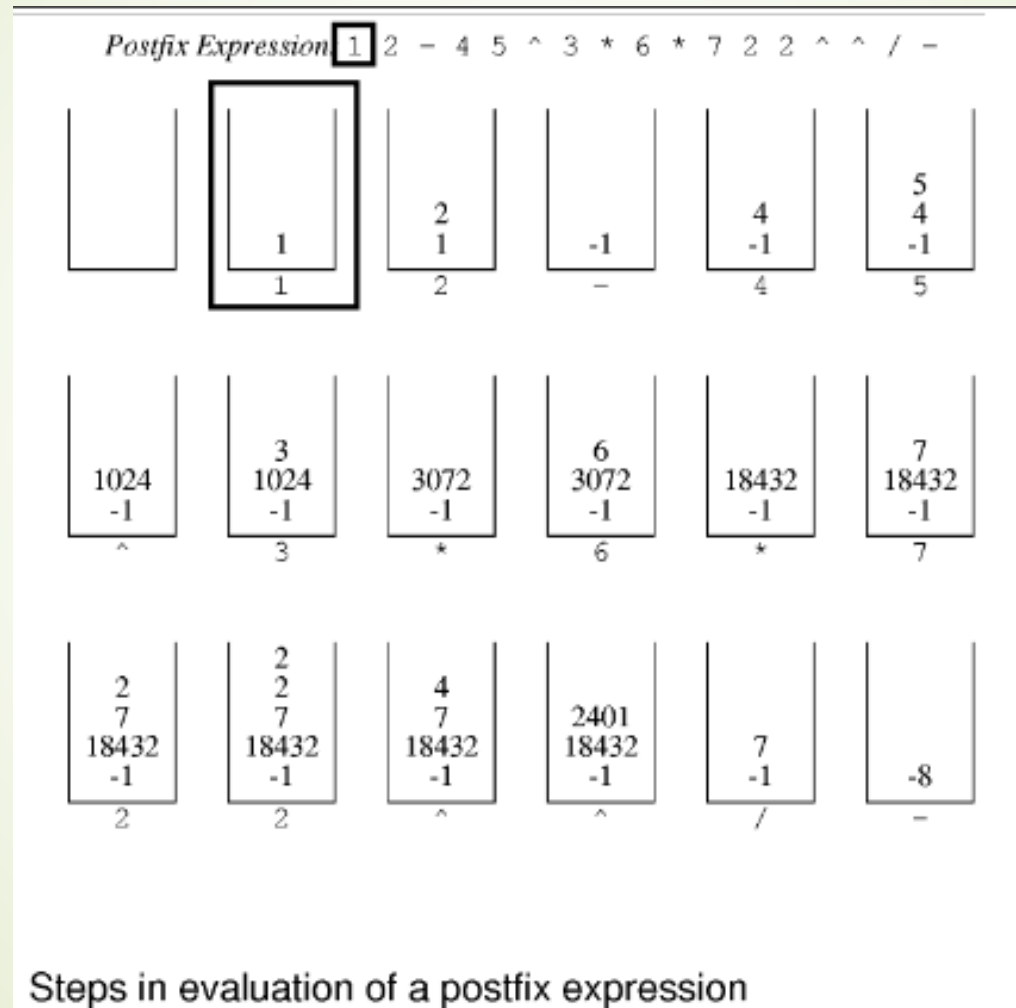
Postfix

$A B C \wedge / D E * A C * - -$
 $((A / (B \wedge C)) - ((D * E) - (A * C)))$


Prefix

$- / A \wedge B C - * D E * A C$

Stack (Yığın)



Stack (Yığın) C# Collection

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Stack st = new Stack();
            st.Push(1);
            st.Push(2);
            st.Push(3);
            st.Push(4);
            st.Pop();

            foreach (Object obj in st)
            {
                Console.WriteLine(obj);
            }
            Console.WriteLine(); Console.WriteLine();
            Console.WriteLine("The number of elements in the stack " +st.Count);
            Console.WriteLine("Does the stack contain the elements 3 "+st.Contains(3));
            Console.ReadKey();
        }
    }
}
```

Stack (Yığın) C# Collection

```
private static bool ParantezKontrol(string Aritmetik)
{
    Stack<string> yigin = new Stack<string>();

    for (int i = 0; i < Aritmetik.Length; i++)
    {
        if (Aritmetik[i] == '(')
        {
            // Okunan her bir sol parantez Stack'e eklenir.
            yigin.Push("(");
        }
        else if (Aritmetik[i] == ')')
        {
            try
            {
                // Okunan her bir sağ parantez için Stack'den bir sol parantez çıkarılır.
                yigin.Pop();
            }
            catch (InvalidOperationException)
            {
                // Aritmetik ifadede açılmamış bir parantez kapatılmaya çalışılmış.
                return false;
            }
        }
    }

    if (yigin.Count != 0) // Açılan bir parantez kapatılmamış.
        return false;
    else // Açılan her bir parantez kapatılmış.
        return true;
}
```

```
static void Main(string[] args)
{
    string Aritmetik1 = "(1+2)+(4*8)+9"; // Parantez kullanımı doğru.
    string Aritmetik2 = "(1+2))+(4*8)+9"; // Açılmamış bir parantez
    kapatılmaya çalışılmış.
    string Aritmetik3 = "((1+2)+(4*8)+9"; // Açılan bir parantez
    kapatılmamış.
    string Aritmetik4 = "(((((((1)))))))"; // Açılmamış bir parantez
    kapatılmaya çalışılmış.

    Console.WriteLine(ParantezKontrol(Aritmetik1)); // TRUE
    Console.WriteLine(ParantezKontrol(Aritmetik2)); // FALSE
    Console.WriteLine(ParantezKontrol(Aritmetik3)); // FALSE
    Console.WriteLine(ParantezKontrol(Aritmetik4)); // FALSE
}
```