YIĞIN YAPILARI

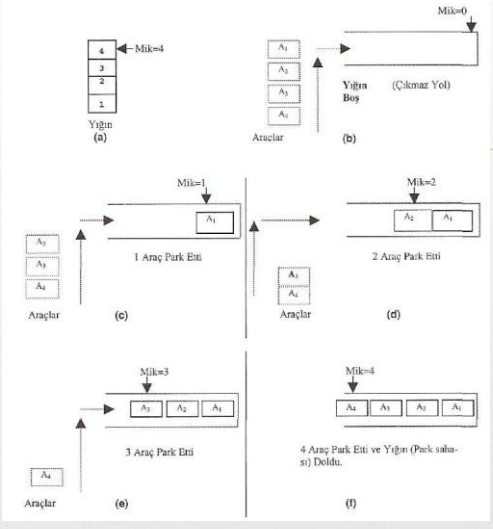
Yığın Yapıları

Yığın, doğrusal bir listedir. Yığına eleman eklemek veya yığından eleman çıkarmak yığını oluşturan listenin bir tarafından yapılır. *"Yığına son giren eleman ilk önce çıkar* (SGİÇ)". Yığın tanımlanırken eleman sayısı ve türü de tanımlanmalıdır. Yığının indis değerini tutan değişkene 'ElemanMiktarı' denilir ve kısaca 'Mik' olarak adlandırılacaktır. Yığına eleman eklenirken 'Mik' değeri arttırılacak, yığından eleman çıkartılırken 'Mik' değeri azaltılacaktır.

Yığın Yapıları

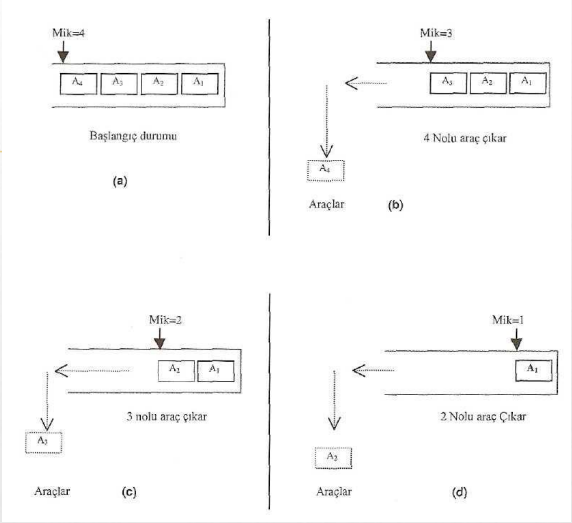
Yığınlar tanımlanırken kapasiteleri tanımlanmalıdır.

Yığın Yapıları



Yığın Yapıları

A2 aracının sokaktan çıkması için yapılacak işlemler. Yığına son giren eleman yığından ilk çıkar.



Yığına Eleman Ekleme İşlemleri

'YiğınaElemanEkle' Algoritması ile yığına aynı anda tek bir eleman eklenir. Eğer yığına eklenmek istenen eleman sayısı birden fazla ise her eleman ekleme istendiğinde 'YığınaElemanEkle' algoritması yeniden çağrılır.

Yığına Eleman Ekleme İşlemleri

Algoritma YığınaElemanEkle (Eleman, Yığın,n,Mik)

//Bu algoritma eleman kapasitesi "n" olan bir yığına gelen elemanı yerleştirir. 'Mik', Yığında bulunan eleman sayısını tutan tamsayı değişken ve 'Eleman' ise yığına konmak üzere gönderilen değeri tutan değişkendir. 'Yığın' değişkeni 'n' elemanlı bir doğrusal listedir.//

1. [Yığın Dolumu?]

```
if Mik >=n  then Begin      //Yığın, dolumu1.' Kontrol edilir. Yığın dolu ise
    Yaz ('Yığın  Dolu');    bu durum mesajla bildirilir ve işlem bitirilir.//
    Adım.3'e  Git;
    End;
```

2. [Yığına eleman ekle]

```
Mik←Mik+1;          //Yığının 'Mik" nolu adresine yeni gelen eleman
Yığın (Mik) ← Eleman;   yerleştirilir.//
```

3.[İşlem bitir]

Dur;

Yığına Eleman Ekleme İşlemleri

Örnek: 20 elemanlı bir yığın tanımlayarak veri ortamından okunan değerleri alt algoritma yardımı ile yığma ekleyen ana algoritmayı yazınız. Algoritma Yığın dolunca bitecektir.

Yığına Eleman Ekleme

```
Algoritma YığınÖrneği;

//Bu algoritma 20 elemanlı bir yığın tanımlayarak veri ortamından okuduğu değerleri 'YığınaElemanEkle'
alt algoritması vasıtasıyla ekler. Algoritmada kullanılan 'n' ve 'Mik' değişkenleri tam sayı değişkenlerdir.
'n' değişkeni yığının kapasitesini, 'Mik' değişkeni de yığına eklenen en son elemanın adresini tutar. 'Veri',
ortamdan değer okunan değişkendir. 'Yığın' değişkeni 'n' elemanlı doğrusal bir listedir.//

1. [İlkleme]
   n←20;              //Bu adımda yığın tanımlanarak kapasitesi belirlenir. Ayrıca ilk atamalar yapılır.//
   Yığın[1:n];
   Mik←0;

2. [Veri Oku]
   Oku(Veri);         //Veri ortamından veri okunur.//

3. [Alt Algoritma Çağır.]
   YığınaElemanEkle(Veri,Yığın,n,Mik);

4. [Yığını Kapasitesini Kontrol Et]
   If Mik≥ n then Adım.6'ya git;

5. [İşlem Tekrarla]
   Adım.2'ye git;

6. [İşlem Bitir.]
   Dur;
```

Algoritma-3.2.2. Yığına ekleme algoritmasının kullanılması

Yığından Eleman Çıkarma İşlemi

Daha önceden de açıklandığı üzere yığın son giren eleman ilk çıkar mantığına göre çalışan doğrusal bir listedir. Bu nedenle Yığma en son eklenen eleman en önce çıkarılır. Bu işlemi yapan algoritma aşağıda verilmiştir.

Yığından Eleman Çıkarma İşlemi

```
Algoritma YığındanElemanÇıkar(Eleman,Yığın,Mik)

//Bu algoritma bir yığında bulunan elemanlardan 'Mik' değişkeni ile işaretli olan elemanı yığından çıkarır.
Daha sonraki işlemler için gerekli atamaları yapar. Yığından çıkarılan eleman 'YığındanElemanÇıkar' algo-
ritmasını çağıran algoritmaya gönderilir. Algoritmada kullanılan 'Mik' değişkeni tamsayı bir değişken olup
çıkarılacak elemanı gösterir. 'Eleman' değişkeni ise yığında saklanan elemanların tipinde bir değişkendir.
'Yığın' değişkeni 'n' elemanlı doğrusal bir listedir. //

1. [Yığın Boşmu?]
   if Mik≤0 then Begin              //Yığın boşmu? Kontrol edilir. Yığın boşsa durum me-
                 Yaz(Yığın Boş);    sajla bildirilir ve işlem bitirilir.//
                 GeriDön;
                 End;

2. [Eleman çıkar]
   Eleman←Yığın(Mik);    //Yığından bir eleman çıkarılır. //

3. ['Mik' değerini bir azalt]
   Mik←Mik-1;

4. [İşlem bitir]
   Dur;
```


Yığından Eleman Çıkarma İşlemi

'YığındanElemanÇıkar' algoritması yığından eleman çıkarılmak istendiğinde bir başka algoritma tarafından çağrılabilir. 'YığındanElemanÇıkar' algoritması her işleyişinde yığından tek eleman çıkarır.

Yığın Kullanım Alanları

Yığınların kullanım alanları oldukça fazladır. Bunlar;

- alt program çağırma
- özyineli (rekursif) algoritma konuları seçilmiştir.

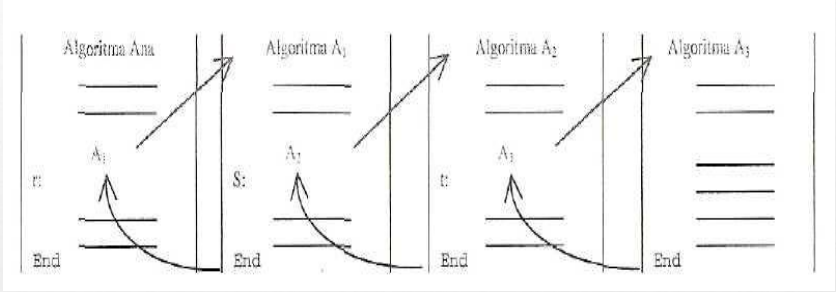
Alt Algoritma Çağırma

Ana algoritma 'A1' isimli alt algoritmayı çağırıyor. 'A1' alt algoritmanın işlevi bitince ana algoritmadaki dönüş adresi 'r:' olacaktır. İşlem akışı 'A1' alt algoritmasına geçince, geri dönüş adresi yığına konur. 'A1' isimli alt algoritma 'A2' isimli alt algoritmayı çağırmaktadır. Yine 'A2' isimli alt algoritma işlemi bitince işlem 'A1' isimli alt algoritmaya geri gelmesi ve 's:'adresinden itibaren devam etmesi gerekir buradaki 's:' gösterilen adresin yine yığında saklanması gerekir.

Alt Algoritma Çağırma

Yine benzer şekilde 'A2' isimli alt algoritma 'A3' isimli alt algoritmayı çağırmakta ve işleme devam etmektedir. Burada dikkat edilirse ilk önce 'A3' isimli alt algoritmanın çalışması bitecek sonra sırası ile 'A2', 'A1' ve ana algoritmanın çalışması bitecek ve işletim sistemine geri dönülecektir. Bu noktadan hareketle bir algoritma içinde yapılan alt algoritma çağırma işlemleri sonunda işlem akışının devam edebilmesi için dönüş adreslerinin yığında saklanması gerekir.

Alt Algoritma Çağırma



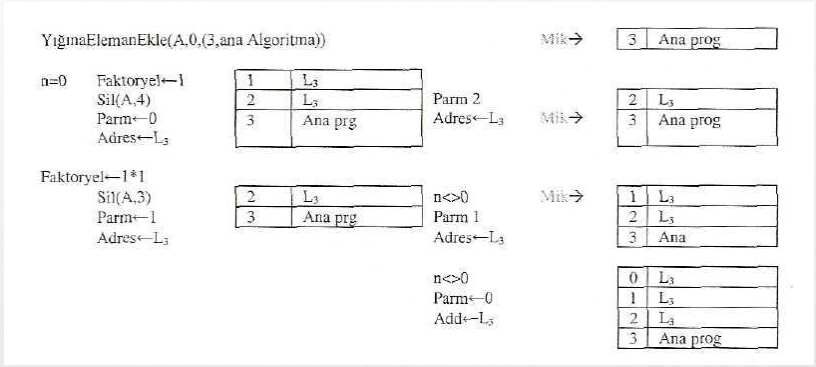
Özyineli (Rekürsif) Algoritma

Bir algoritmanın kendi kendini çağırması olarak bilinen özyineli algoritmalar en çok kullanılan yapılardan bir tanesidir. Bu tür algoritmalarda algoritma kendi kendisini parametreleri ile birlikte çağırarak işlem yapar.

Özyineli (Rekürsif) Algoritma

Kendisini her çağırışta algoritma ismi ve gönderilen parametreler yığında saklanır. Bu algoritmaya örnek olarak faköriyel hesaplama verilebilir. Aşağıdaki örnekte faktöriyel hesaplarken özyineli algoritmanın yığın kullanımı verilmiştir.

Özyineli (Rekürsif) Algoritma



İfadelerin Değerlendirilmeleri

Üst düzey programlama dilleri geliştirilirken bu alanda çalışan bilim adamlarının karşılaştıkları zorluklardan biriside üst düzeyde yazılan aritmetiksel ifadelerin makine kodunda nasıl değerlendirilip işleneceği idi. Örneğin,

X← A/B**C+D*E-A*C

gibi bir ifadenin değerlendirilmesinde birden fazla olasılık ortaya çıkar.

İfadelerin Değerlendirilmeleri

Bu ifadeyi işlem sırasına göre parantezlere de alsak, yine de bu ifadenin değerlendirilmesi için makine dili kodunun üretilmesi oldukça zor ve karmaşıktır. İfadeler 'değişken', ‘sabit' ve 'operatör'lerden oluşurlar. Yukarıda verilen ifade beş adet değişkenden oluşmaktadır. Bunlar ‘A', 'B', 'C’, 'D' ve 'E‘ dir.

İfadelerin Değerlendirilmeleri

Bu Değişkenler tek karakterden oluşmuşlardır. Gerçekte, değişkenler geçerli olan herhangi bir değişken veya sabit isminden oluşabilirler. Bu durum kullanılan programlama dili ile ilgilidir.

İfadelerin Değerlendirilmeleri

Değişkenler arasında operatörler (işlemciler) bulunur. İfadeler nasıl yazılırlarsa yazılsınlar sonuçlarının doğru olması gerekir. Bunun için ifadelerde kullanılan operatörlerin işleme alınma önceliklerinin bilinmesi önemlidir.

İfadelerin Değerlendirilmeleri

Aslında, istenirse, işlemler parantez içine alınarak öncelik verilebilir; ancak bu fazladan işlem karmaşık ifadelerde zorluklar yaratır. Bu kısa açıklamadan sonra aritmetik operatörlerin değişkenlerle bulunabilecekleri durumları incelemek gerekir.

İfadelerin Değerlendirilmeleri

Genel anlamda herhangi bir ifadeyi işlemek için iki yöntem vardır; bunlar:

a. Soldan sağa veya sağdan sola değerlendirme yöntemi; Bu yöntemde ifadeler soldan sağa veya sağdan sola doğru işlenirler.

b. Öncelikler yöntemi; Bu yöntemde operatörlere öncelik değerleri verilir. Değeri yüksek olan operatör önce işlenir

İfadelerin Değerlendirilmeleri

Öncelik değerler şöyledir:

- ** Birinci öncelikle işlenir.
- *, / İkinci öncelikle işlenirler.
- +, - Üçüncü öncelikle işlenirler.

Aritmetik İfade Biçimleri

Aritmetik ifadeler yazıldıkları şekilde çözülemezler. Aritmetik operatörlerinin arasında işlem öncelikleri vardır. Örneğin parantez içindeki ifadelerin öncelikleri vardır, Bir başka örnek çarpma işlemi toplama işleminden önce yapılmalıdır. Aritmetik ifadelerin doğru işlenmeleri için yöntemler geliştirmek gerekir.

Operatörler Arada Biçimi (Opard)

(A+B)*(C-D)

Bu yapıda operatörler her zaman değişkenlerin arasındadır. Zaten temel aritmetik yapıda bunu gerektirir

Operatörler Önde Biçimi

Bu yöntem iki alt yöntem halinde incelenecektir. Bu yöntemlerin hepsinde operatörler değişkenlerin önünde yer alırlar; ancak aralarında az da olsa farklılıklar vardır.

Operatörler Önde Biçimi

Temel Biçim:

Bu yöntemde operatörler değişkenlerden önce yer alırlar.

Operatörler Önde Biçimi

Örnek: (A+B)*(C-A) Verilen bu ifade normal matematiksel bir ifadedir. Bu ifadeyi 'Temel Biçin:' ifadeye çevirmek için operatörler değişkenlerin önüne, bir başka deyişle baş tarafın: alınırlar. Operatörlerin yerlerine ',' konulur. Verilen örnek ifadede operatörlere yapılar işlemleri adım adım gösterelim:

(A+B)→+(A,B) + operatörü başa alındı yerine ',' kondu

(C-A) → -(C,A) - operatörü başa alındı yerine ',' kondu

(A+B)*(C-A) → *(+(A,B)-(C,A)) Son durum

Operatörler Önde Biçimi

Cambridge Biçimi:

(*(+AB)(-CA)

Bu yöntemde verilen ifade önce 'Temel Biçim'e çevrilir. Sonra değişkenlerin arasındaki virgüller de atılır.

Operatörler Önde Biçimi

Yalın Biçim:

*+AB-CA

Bu yöntemde verilen ifade önce 'Temel Biçim'e sonra *'Cambridge* Biçimi'ne çevrilir ve daha sonra ifadede bulunan parantezlerde atılır.

Operatörler Sonda Biçimi (OpSonda)

"Operatörler arada' ve 'operatörler önde' yöntemlerinin aksine 'Operatörler Sonda' yönteminde operatörler değişkenlerin en sonuna gönderilirler.

(A+B)*(C-A)

((A,B)+,(C,A)-)*: Operatörler sona geldi parantez ve virgül kalktı.

AB+CA-*

Operatörler Sonda İfade Biçimi

Bilgisayarda yazılan programlarda kullanılan ifadelerin değerlendirilmesi doğrudan yapılamaz. Aritmetiksel ifadelere 'operatörler arada ifadeler' denilir. Bu ifadeler önce 'operatörler sonda ifade' biçimine dönüştürülürler ve sonra değerlendirilirler. Bu bölümde bu işlemleri yapan algoritmalar incelenecekler.

Operatörler Sonda İfade Biçimi

Algoritmaları incelemeye başlamadan önce, ifadelerin dönüştürülmelerinde kullanılacak olan operatörlerin yığın içinde ve yığın dışındaki birbirleri arasında bulunan işlem öncelik değerlerini bilmek gerekir.

Operatörler Sonda İfade Biçimi

Operatörlerin işlem öncelikleri

Simge	Yığın içindeki öncelik değeri (YÖN)	Yığına gelirken sahip olduğu öncelik değeri (GÖN)
)	-	-
**	3	4
*,/	2	2
+,-	1	1
(	0	4

Operatörler Sonda Algoritması

Verilen Operatörler arada ifadeyi değerlendirmeye almadan önce operatörler sonda ifadeye çeviren algoritma aşağıda verilmiştir. Algoritmanın işlem adımlarında yaptıklarını adım adım açıklamakta yarar vardır;

Operatörler Sonda Algoritması

- Adım 1: Bu adımda algoritmada ilk işlemler yapılmaktadır; bunlar; OpArd ifadenin bittiğini belirten ve OpArd'ın son karakteri olması istenen + ∞'un OpArd'a eklenmesi, Yığının indis değeri olan Mik değişkenine ilk değer atanması ve yığının ilk elemanı olacak -∞ 'un yığına atanmasıdır.
- Adım 2: Bu adımda OpArd ifadesinden yeni bir eleman alınarak değerlendirmeye verilir.

Operatörler Sonda Algoritması

- Adım 3: Bu adımda OpArd ifadenin bitip bitmediği kontrol edilir. Eğer ifade bitmiş ise yığında bulunan tüm değişkenler yığından çıkartılırlar. İşlemin bittiği belirtilerek algoritma sonlandırılır.
- Adım 4: Bu adımda işleme alınan elemanın değişken olup olmadığı kontrol edilir. Eğer eleman, değişken ise yığına atanmayıp doğrudan çıktıya gönderilir. Sonra işlem akışı bir sonraki elemanın değerlendirilmesi için Adım 2'ye yönlendirilir.

Operatörler Sonda Algoritması

- Adım 5: Bu adımda değerlendirmeye alınan elemanın ')' olup olmadığı kontrol edilir. Eğer eleman ')' ise yığında bulunan elemanlar '(' olmadığı sürece yığından çıkartılıp yazıcıya gönderilirler, '(' e ulaşılınca yığından çıkartılır; ancak işleme alınmaz. İşlem akışı bir sonraki elemanın değerlendirilmesi için Adım 2'ye yönlendirilir.

Operatörler Sonda Algoritması

- Adım 6: Bu adımda değerlendirmeye alınacak elemanın artık bir operatör olduğu belirlenir. Bu elemanın yığına girme şartı sağlanana kadar yığından eleman çıkartılır. Koşul sağlanınca eleman yığma atanır ve bir sonraki elemanın değerlendirilmesi için işlem akışı Adım 2'ye yönlendirilir.

Operatörler Sonda Algoritması

```
Algoritma OperatörlerSonda(OpArd)

// Bu algoritma Operatörler Arada 'OpArd' ifadeyi Operatörler Sonda 'OpSonda' ifadeye çevirir.
SonrakiEleman bir algoritma olup 'OpArd' ifadede bulunan elamanları tek tek getirir. 'Yığın[1:n]' bu tanım
operatörleri tutacağımız 'n' elemanlı bir yığındır. Yığının ilk elemanı olan '-∞' un yığın içindeki öncelik
değeri –1'dir. 'YÖN', 'GÖN' ifadeleri birer adet fonksiyon olup operatörlerin yığın içinde ve yığın dışındaki
işlem öncelik değerlerini bulurlar. //

1. [İlk İşlemler]
   Mik←1;
   Yığın(mik)←'-∞';
   OpArd←OpArd+'∞';

2. [Yeni Eleman Al]
   x←SonrakiEleman(OpArd);          // 'OpArd' ifadeden bir eleman al.//
```


Operatörler Sonda Algoritması

```
3. [x=∞ mu?]
   if x='∞' then  Begin
                  While Mik>1 do
                              YığındanElemanÇıkar(Eleman,Yığın(Mik);
                              Yaz(Eleman);
                              Repeat
                  Yaz ('∞');
                  GeriDön;
                  End;

4. ['x' bir değişkenmi?]
   If x = değişken then Begin
                        Yaz(x);
                        Adım.2'ye git;
                        End;

5. ['x', ')' mi?]
   If x=')' then Begin
                 While Yığın(Mik)≠'(' do
                                YığındanElemanÇıkar(Eleman,Yığın(Mik)
                                Yaz(Eleman);
                                Repeat
                 YığındanElemanÇıkar(Eleman,Yığın(Mik);
                 Adım.2'ye git;
                 End;

6. //Sonraki eleman bir operatördür.//
   while YÖN(yığın(Mik))≥ GÖN(x) do
                          YığındanElemanÇıkar(Eleman,Yığın(Mik);
                          Yaz(Eleman);
                          Repeat
   YığınaElemanEkle(x,yığın,n,Mik);
   Adım.2'ye git;

7. [İşlem Bitir]
   Dur;
```

Algoritma-3.6.2.1. Operatör arada ifadeyi operatör sonda ifadeye çevirir

Operatörler Sonda Algoritması

```
Algoritma OperatörlerSonda(İ)

// Bu algoritma 'i' ile verilen 'OpArd' ifadeyi 'OpSonda' ifadeye çevir. 'i' ile verilen ifadenin son karakteri
'∞' karakteri olup buna ulaşılınca ifade bitmiş olacaktır. Yığında başlangıç değeri olarak 'Mik=1' yapılacak
ve ilk eleman olarak yığına '-∞' konulacaktır. SonrakiEleman bir algoritma olup 'OpArd' ifadede bulunan
elamanları tek tek getirir. Yığın[1:n] bu tanım operatörleri tutacağımız 'n' elemanlı bir yığındır. Yığının ilk
elemanı olan '-∞' un yığın içindeki öncelik değeri –1'dir. 'YÖN', 'GÖN' ifadeleri birer adet fonksiyon
olup operatörlerin yığım içinde ve yığın dışındaki işlem öncelik değerlerini bulurlar. //

1. [İlk İşlemler]
   Yığın(1) ←'-∞';
   Mik←1 ;
```


Operatörler Sonda Algoritması

```
2. [Verilen 'i' İfadesi Bitene Kadar İşle]
   Döngü          //Sonsuz döngü içine girilir//

3. [Yeni Elema Al]
   X←SonrakiEleman(İ) ;    //'İ' ile verilen 'OpArd' ifadeden bir eleman al.//

4. [Elemanı Değerlendir]
   CASE
     :X='∞' Begin
          While Mik>1 do
                    Yaz(Yığın(Mik));    //'X':∞ demek 'OpArd' ifade bitmiş demektir.
                    Mik←Mik-1;          O zaman yığıntaki tüm elemanları yazıcıya gön-
                    Repeat              der.//
          Yaz (∞);
          GeriDön;
          End;
     :X is an operand :Yaz(x);          //Değişkenler doğrudan yazdırılır.//
     :X ='(':Begin
          While Yığın(Mik)≠ '(' do                    //'(' e kadar yığın tan
                                  Yaz(yığın(Mik));    eleman çıkar.//
                                  Mik←Mik-1;
                                  Repeat
          Mik←Mik-1;    //Burada '(' i Yığından atıyor.//
          End;
     Else :    //Sonraki eleman bir operatördür.//
     while YÖN(yığın(Mik))≥ GÖN(x) do
                                  Yaz(yığın(Mik));
                                  Mik←Mik-1;
                                  Repeat
     YığınaElemanEkle(x,yığın,n,Mik);
     End {CASE}
   Sonsuz
5. [İşlem Bitir]
   End; {OpSonda}
```

Algoritma-3.6.2.1.

Operatörler Sonda İfadeyi Değerlendirme İşlemi

Verilen normal 'Operatörler arada ifade' Algoritma ile 'operatörler sonda' ifadeye dönüştürüldükten sonra elde edilen yeni ifade artık değerlendirilmeye alınabilir. İfade soldan sağa doğru değerlendirme yöntemine göre değerlendirmeye alınacaktır.

Operatörler Sonda İfadeyi Değerlendirme İşlemi

İfadenin değerlendirilmesinde operatöre rastlanana kadar karşılaşılan değişken değerleri okunarak yığma atanacaktır. Operatör ile karşılaşılınca yığından iki tane değer çıkarılacak operatör ile işlendikten sonra elde edilen yeni değer tekrar yığma atanacak. Bu işlem ifade bitene kadar devam edecek ve ifade bittiğinde yığının birinci elemanı tek değer olacaktır.

Operatörler Sonda İfadeyi Değerlendirme İşlemi

```
Algoritma OperatörlerSondaHesapla(İ)
//Bu algoritma verilen 'OpSonda' ifadeyi değerlendirir ve sonucu hesaplar.//
1. [İlkleme]
   Mik←0;
2. [Değerlendir]
   Döngü
   X←SonrakiEleman(i);
   Case
     X = '∞':GeriDön
             :x  değişkeni bir operatör, YığınaElemanEkle(x,Yığın,n,Mik)
             :Else      Yığından yeteri kadar değişkeni çıkar; x değişkeni içinde gelen operatörü
                        kullanarak hesap işlemini yap, sonucunu tekrar yığına yerleştir.
     End; {Case }
   Sonsuz
3. [İşlem Bitir]
   Dur;
```

Algoritma-3.6.3.1.

Operatörler Sonda İfadeyi Değerlendirme İşlemi

Algoritma-3.6.3.1'i biraz daha somutlaştıralım ve aşağıdaki algoritmayı yazarak değerlendirme işlemini daha somut işleyelim.

Operatörler Sonda İfadeyi Değerlendirme İşlemi

```
Algoritma OperatorlerSondaDeğerlendir(OpSonda)
// Bu algoritma verilen 'OpSonda' ifadeyi değişkenlere değerler okuyarak hesaplar ve sonucunu istenen yere
gönderir. n elemanlı bir yığın, 'Veri', 'veri1' ve 'veri2' isimli aynı türden üç adet değişken kullanılmıştır.//
1. [İlk İşlemler.]
   YığınBoşalt(1:n);
   Mik=0;
2. [Verilen 'OpSonda' İfadesini değerlendir]
   Eleman=SonrakiEleman(İşSonda);
3. ['OpSonda' ifadesi bitimi?]
   if Eleman= ∞ then Begin
                     YığındanElemanÇıkar(veri1,Yığın,Mik);
                     Yaz(Eleman);
                     GoriDön;
                     End;
4. [Eleman bir değişkenmi?]
   if Eleman=değişken then Begin
                           Oku(veri);
                           YığınaElemanEkle(veri,Yığın,n,Mik);
                           End;
   Adım.2'ye git;
5. [Eleman bir operatör mü?]
   if Eleman=operatör then Begin
                           YığındanElemanÇıkar(veri1,Yığın,Mik);
                           YığındanElemanÇıkar(veri2,Yığın,Mik);
                           veri=veri2 Eleman veri1;
                           YığınaElemanEkle(veri,Yığın,n,Mik);
                           End;
   Adım.2'ye git;
6. [İşlem Bitir.]
   Dur;
```

Algoritma-3.6.3.2.

Dönüştürme Örnekleri

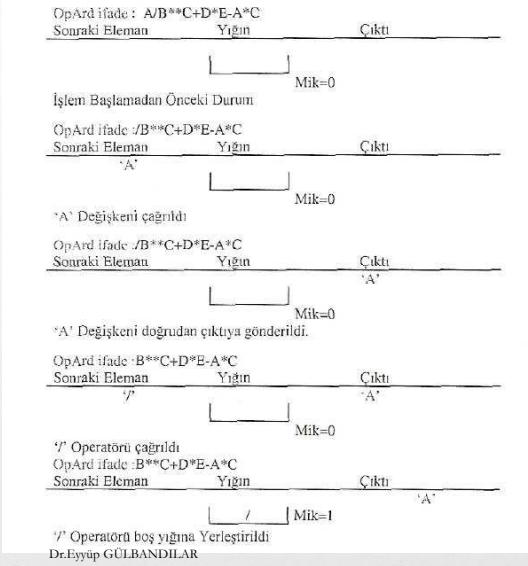
Örnek 1: Operatörler arada olarak verilen A+B*C ifadesini, operatörler sonda ifadeye çeviriniz.

Sonraki Eleman	Yığın	Çıktı
A		A
+	+	A
B	+	AB
*	+*	AB
C	+*	ABC*+

Dönüştürme Örnekleri

Örnek 2: 'OpArd' ifadeden 'OpSonda' ifadeye dönüşü gösterelim.

Dönüştürme Örnekleri



Dönüştürme Örnekleri

OpArd ifade :**C+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
'B'	/	'A'

Mik=1

'B' Değişkeni çağrıldı.

OpArd ifade :**C+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
	/	'A B'

Mik=1

'B' Doğrudan çıktıya gönderildi.

OpArd ifade :C+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
'**'	/	'A B'

Mik=1

'**' operatörü çağrıldı.

OpArd ifade :C+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
	** /	'A B'

Mik=2

Dönüştürme Örnekleri

GÖN(**)≥YÖN(/) dır. '**' yığına yerleştirilir.

OpArd ifade :+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
'C'	** /	'A B'

Mik=2

'C' Değişkeni çağrıldı.

OpArd ifade :+D*E-A*C

Sonraki Eleman	Yığın	Çıktı
	** /	'A B C'

Mik=2

'C' Doğrudan çıktıya gönderilir.

OpArd ifade :D*E-A*C

Sonraki Eleman	Yığın	Çıktı
'+'	** /	'A B C'

Mik=2

'+' Operatörü çağrıldı.

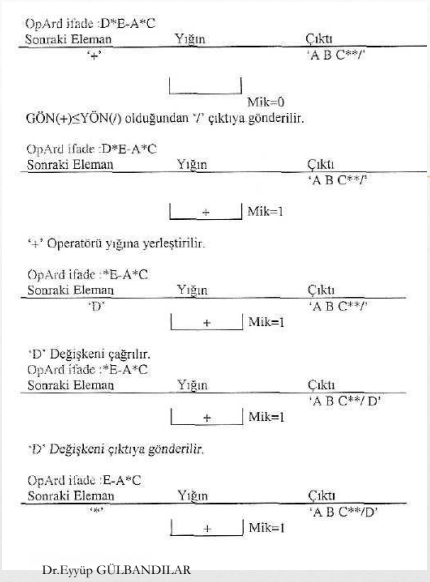
OpArd ifade :D*E-A*C

Sonraki Eleman	Yığın	Çıktı
'+'	/	'A B C**'

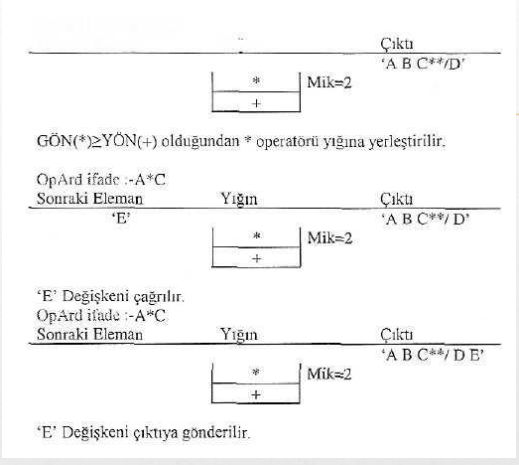
Mik=1

GÖN(+)≤YÖN(**) olduğu için ** çıktıya gönderilir.

Dönüştürme Örnekleri



Dönüştürme Örnekleri



Dönüştürme Örnekleri

OpArd ifade :A*C

Sonraki Eleman	Yığın	Çıktı
'-'		'A B C**/ D E'
	*	Mik=2
	+	

'-' Operatörü çağrılır.

OpArd ifade :A*C

Sonraki Eleman	Yığın	Çıktı
'-'		'A B C**/ D E*'
	+	Mik=1

GÖN(-)≤YÖN(*) oldundan '*' çarpma operatörü çıktıya gönderilir.

OpArd ifade :A*C

Sonraki Eleman	Yığın	Çıktı
'-'		'ABC**/D E*+'
		Mik=0

GÖN(-)≤YÖN(+) olduğundan '+' çıktıya gönderilir.

OpArd ifade :A*C

Sonraki Eleman	Yığın	Çıktı
		'ABC**/DE*+'
	-	Mik=1

Dönüştürme Örnekleri

	Yığın	Çıktı
'A'		'ABC**/D E*+'
	-	Mik=1

'A' Değişkeni çağrılır

OpArd ifade :*C

Sonraki Eleman	Yığın	Çıktı
		'ABC**/DE*+A'
	-	Mik=1

Değişken 'A' doğrudan çıktıya gönderilir.

OpArd ifade :C

Sonraki Eleman	Yığın	Çıktı
		'ABC**/DE*+A'
	*	Mik=2
	-	

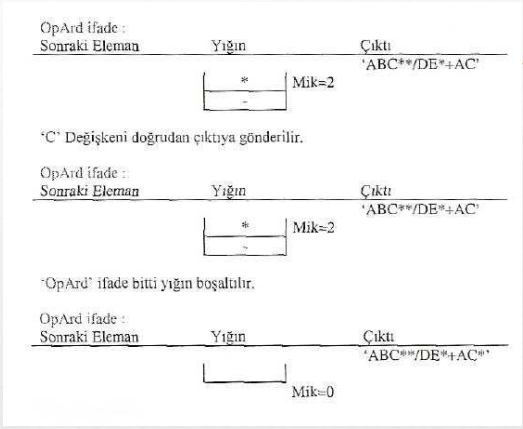
GÖN(*)≥YÖN(-) dir ve '*' Yığına yerleştirilir.

OpArd ifade :

Sonraki Eleman	Yığın	Çıktı
'C'		'ABC**/DE*+A'
	*	Mik=2
	-	

'C' Değişkeni çağrılır

Dönüştürme Örnekleri



Dönüştürme Örnekleri

Bir önceki örneğin tablo halinde gösterimi

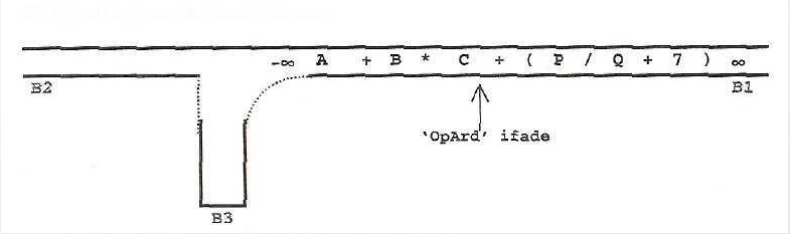
OpArd İfade:A/B**C+D*E-A*C

Sonraki Eleman	Yığın	Çıktı	
A		A	
/	/	A	
B	/	AB	
**	/**	AB	
C	/**	ABC	
+	+	ABC**/	
D	+	ABC**/D	
*	+*	ABC**/D	
E	+*	ABC**/DE	
-	-	ABC**/DE*	
A	-	ABC**/DE*+	
*	-*	ABC**/DE*+A	
C	-*	ABC**/DE*+AC*-	'OpSonda' ifade

Dönüştürme Örnekleri

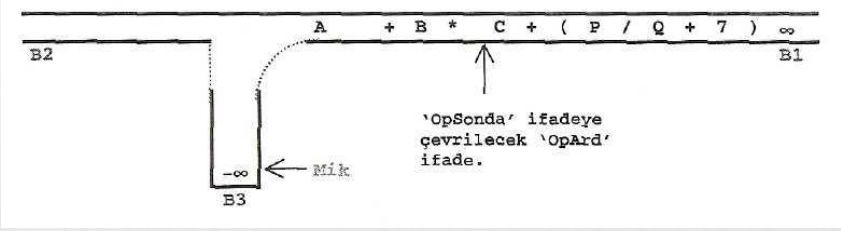
ÖRNEK: Yığından eleman çıkarma kuralı YÖN(AritmetikOperatör) > GÖN(AritmetikOperatör) olacak.

Dönüştürme Örnekleri



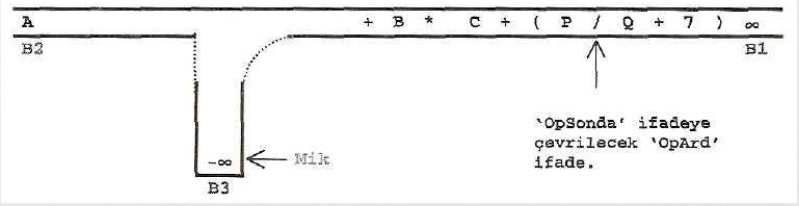
Dönüştürme Örnekleri

- B1 bölgesinde bulunan 'OpArd' ifade soldan sağa doğru işleme alınır. İlk karakter '-∞' dur. Bu değer ilk eleman olarak yığma yerleştirilir.



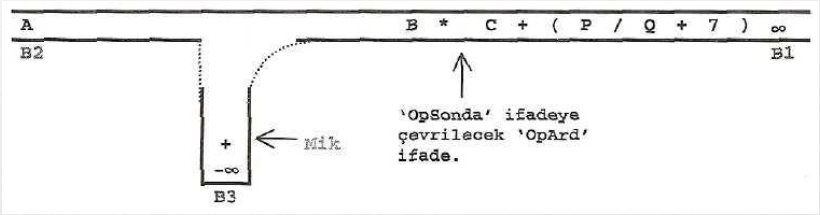
Dönüştürme Örnekleri

- B1 bölgesindeki ifadenin ikinci elemanı değişken, yani 'A' olup doğrudan B2 bölgesine aktarılır.



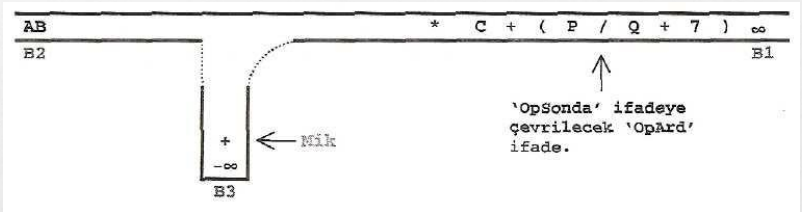
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '+' olup B3 bölgesindeki yığma yerleştirilir.



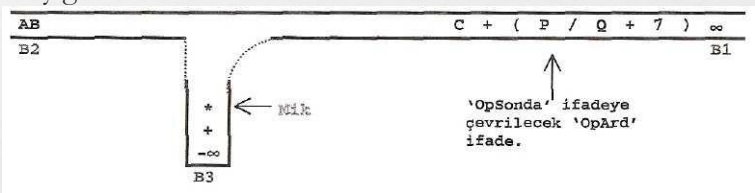
Dönüştürme Örnekleri

- B1 bölgesindeki 'OpArd' ifadenin sonraki elemanı değişken 'B' olup doğrudan B2 bölgesine aktarılır.



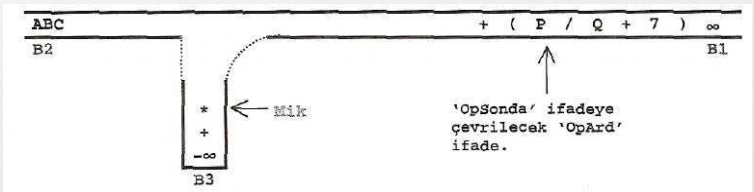
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '*'dır. GÖN(*)=2 dir. YÖN(+)=l'dir. GÖN(*)>=YÖN(+) olduğundan '*' operatörü B3 yığına eklenir.



Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı 'C' olup doğrudan B2 bölgesine aktarılır.

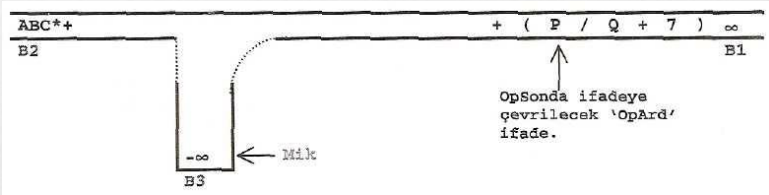


Dönüştürme Örnekleri

- İfadenin bir sonraki eleman +'dir. GÖN(+)=1'dir. YÖN(*)=2'dir. GÖN(+)=<YÖN(*) olduğundan, '*' operatörü yığından çıkarılıp B2 bölgesine gönderilir.

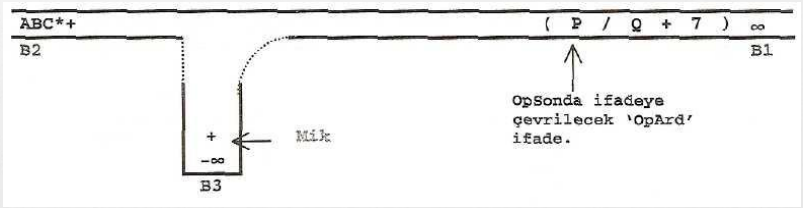
Dönüştürme Örnekleri

- 'OpArd' ifadede bulunan operatör '+'dır. GÖN(+)=1'dir. Yığında bulunan '+'nın YÖN(+)=1'dir. GÖN(+)=<YÖN(+) olduğundan yığında bulunan '+' B2 bölgesine gönderilir.



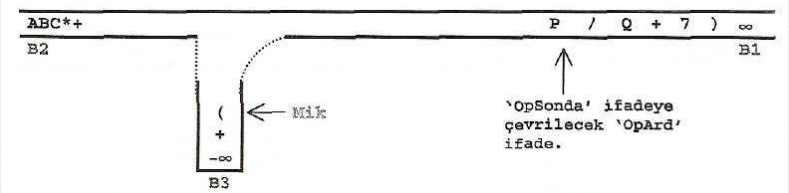
Dönüştürme Örnekleri

- Yığın boşaltıldı. Bu adımda 'OpArd' ifadede bulunan '+' değeri yığma yerleştirilir.



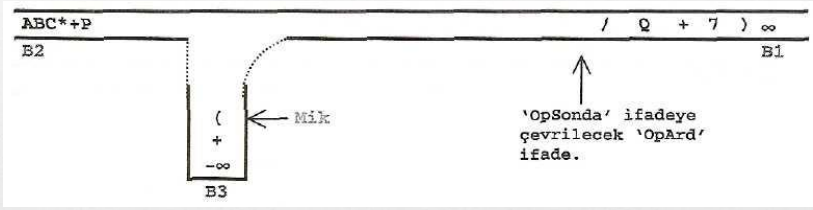
Dönüştürme Örnekleri

- 'OpArd' ifadede bulunan bir sonraki eleman '('dir, GÖN('(')=4'dür ve yığma yerleştirilir.



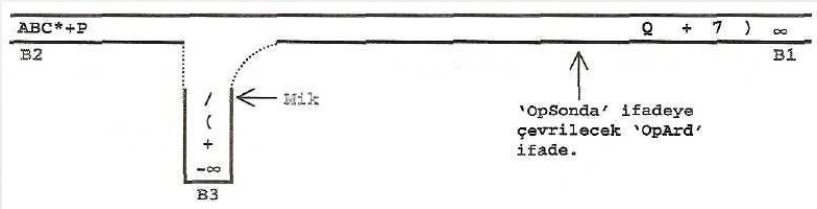
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı 'P' olup doğrudan B2 bölgesine aktarılır.



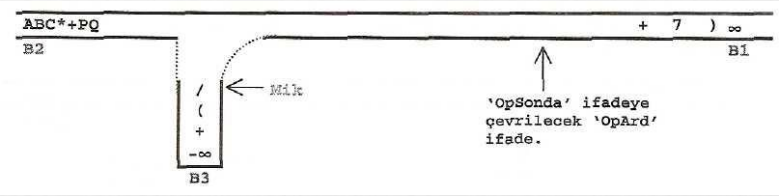
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '/' dür. GÖN(/)=2'dir. Yığın'ta bulunan '(' YÖN('(')=0'dır. 'OpArd' ifadede bulunan '/' elemanı yığına yerleştirilir.



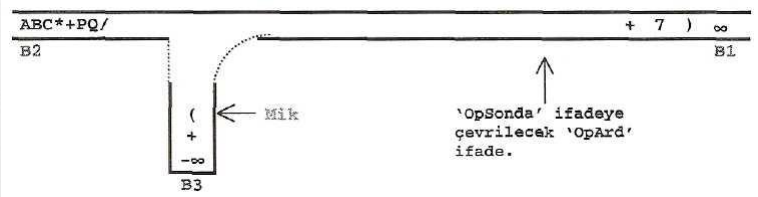
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı ‘Q' olup doğrudan B2 bölgesine gönderilir.



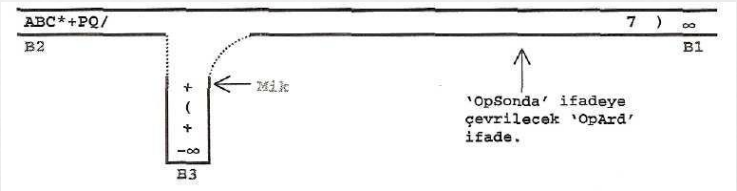
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '+'dır. GÖN(+)=1’dir. Yığında bulunan ‘/' nün YÖN(/)=2'dir. Yığında bulunan ‘/' operatörü yığından çıkarılarak B2 bölgesine gönderilir.



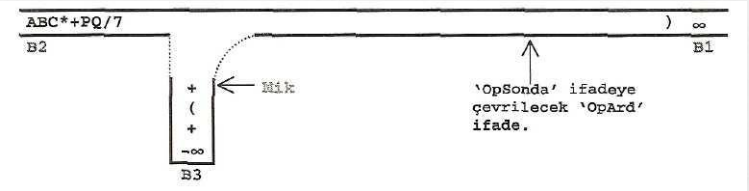
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '+'dır. GÖN(+)=1'dir. Yığında bulunan '(' nün YÖN('(')=0'dir. '+' operatörü Yığına yerleştirilir.



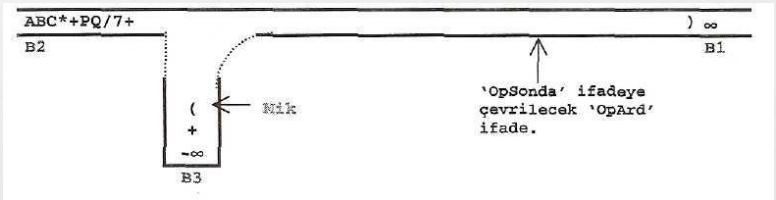
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '7' olup doğrudan B2 bölgesine gönderilir.



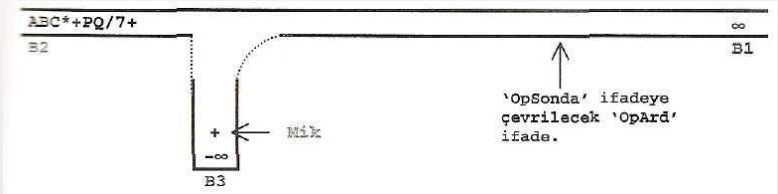
Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı ')'dir. Bu elemana rastlanınca yığında bulunan elemanlar '(' kadar yığından dışarı çıkarılır ve B2 bölgesine gönderilir. Ve ')' elemanı işleme alınmaz.



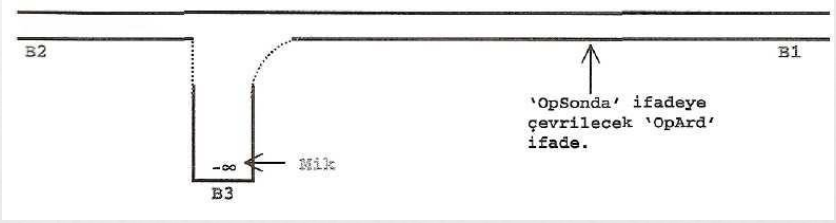
Dönüştürme Örnekleri

- Daha sonra '(' de yığından çıkarılır.



Dönüştürme Örnekleri

- 'OpArd' ifadenin bir sonraki elemanı '∞'dur. Bu elemana ulaşınca 'OpArd' ifade bitmiştir ve yığın boşaltılarak operatörler B2 bölgesine gönderilir. ∞ elemanı işleme alınmaz.



Dönüştürme Örnekleri

- Sonuç olarak elde edilen 'OpSonda' ifade **ABC*+PQ/7++** ifadesidir.

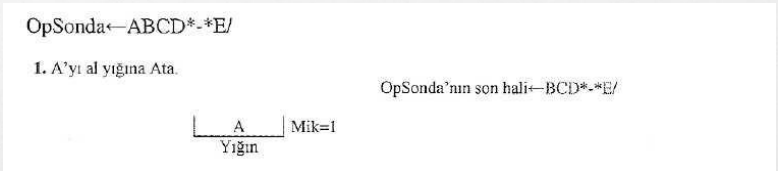
Operatörler Sonda İfadeleri Değerlendirme Örnekleri

- **Örnek 1:** ABCD*-*E/ 'OpSonda' ifadesi üretildi. Şimdi bu ifadeye sayısal değerler vererek sonuç bulunur.

Operation	OpSonda
T1←C*D	ABT1-*E/
T2 ←B-T1	AT2*E/
T3 ←A*T2	T3E/
T4 ←T3/E	T4

Operatörler Sonda İfadeleri Değerlendirme Örnekleri

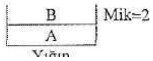
- Şimdi görsel olarak bu işlemleri gösterelim.



Operatörler Sonda İfadeleri Değerlendirme Örnekleri

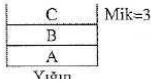
2. B'yi al yığına Ata.

OpSonda'nın son hali←CD*-*E/



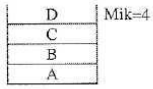
3. C'yi al yığına Ata.

OpSonda'nın son hali←D*-*E/



4. D'yi al yığına Ata.

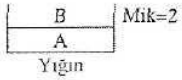
OpSonda'nın son hali←*-*E/



Operatörler Sonda İfadeleri Değerlendirme Örnekleri

Yığından 2 eleman çıkart birbirleri ile çarp sonucunu T1'e ata.

OpSonda'nın son hali←-*E/

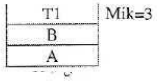


T1←C*D

Operatörler Sonda İfadeleri Değerlendirme Örnekleri

6. T1'i yığına ata.

OpSonda'nın son hali←-*E/



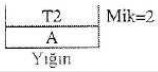
7. Yığından 2 eleman çıkart ve çıkartma işlemini uygula sonucu T2 ye ata.

OpSonda'nın son hali←*E/

T2←B-T1

8. T2 değerini yığına ata

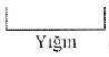
OpSonda'nın son hali←*E/



Operatörler Sonda İfadeleri Değerlendirme Örnekleri

9. Yığından 2 eleman çıkart, çarp ve sonucunu T3'e ata.

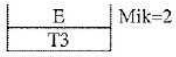
OpSonda'nın son hali←E/



T3←T2*A

11. E değerini yığına at.

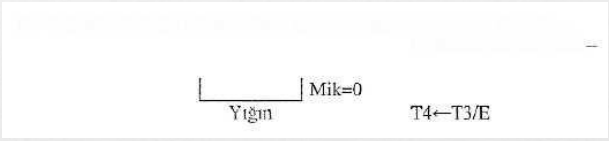
OpSonda'nın son hali←/



Operatörler Sonda İfadeleri Değerlendirme Örnekleri

Yığından 2 değer çıkart, bölme işlemini uygula sonucunu T4'e ata.

OpSonda'nın son hali←



Operatörler Sonda İfadeleri Değerlendirme Örnekleri

13. T4 değerini yığına ata.