

Matematiksel İfadeler

C ve Javada Öncelik Sırası

1. - (eksili sayıları belirten - işareti)
2. * - %
3. + -
4. && ||
- 5.

Matematiksel İfade Veri Yapısı

```
Public class Ornek{
    Int tip;
    Int islenen;
    Char islem;
    Int oncelik;
    Public Ornek(int islenen){
        This.tip=0;
        This.islenen=islenen;
    }
    Public Ornek(int islem){
        This.tip=1;
        This.islem=islem;
    }
    switch (islem){
        case '(' :oncelik = 0;
            break;
        case '+':
        case '-':oncelik = 1;
            break;
        case '*':
        case '/':oncelik = 2;
            break;
        case ')':oncelik = 3;
            break;
    }
}
```

Tip matematiksel ifadedeki her bir elemanın işlenen mi ,işlem mi olduğunu göstermektedir.tip 0 sa eleman bir işlenen,1 se bir işlemdir.Eleman bir işlenirse değeri işlenen alanında tutulurken,eleman bir işlemse değeri işlem alanında tutulmaktadır.Eleman bir işlemse öncelik değeri de oncelik alanında tutulmaktadır.+ ve - işlemlerinin önceliği en düşük olduğundan öncelikleri 1, * ve / işlemlerinin öncelikleri + ve - den yüksek olduğu için öncelikleri 2,dir.Parantez açma işleminin önceliği en düşük ve 0 iken ,parantez kapama işleminin önceliği en yüksek ve 3 tür.

Matematiksel İfadeler

$$a/b - c + d * e - a * c$$

- İşlemler: /, -, +, *, -, *
- İşlenenler: a, b, c, d ve e

Infix = operatör arada(a+b)
Postfix=operatör sonda(ab+)
Prefix=operatör basta(+ab)

Infix : paranteze ihtiyac duyar
Postfix : operatör sonda
Prefix : soldan>>sağa

İşlem Önceliği

Parantez
Üs Alma
Çarpma/Bölme
Toplama / Çıkarma

Arka Gösterim

Bu gösterim biçiminde işlem işaretleri işlenenlerden daha sonra gelir.

$ab/c - de * + ac * -$ arka gösteriminin değerinin hesaplanması aşağıda gösterilmiştir.

$T_1 = a / b$	$T_1 c - d e * + a c * -$
$T_2 = T_1 - c$	$T_2 d e * + a c * -$
$T_3 = d * e$	$T_2 T_3 + a c * -$
$T_4 = T_2 + T_3$	$T_4 a c * -$
$T_5 = a * c$	$T_4 T_5 -$
$T_6 = T_4 - T_5$	T_6

Arka gösterimi verilen bir ifadenin değerini hesaplayan algoritma (1)

```
int hesapla(Ornek[] ifade){
    int i;
    Ornek e, e1, e2, s;
    Cikin c = new Cikin(100); //kullanılacak çıkın
    for (i = 0; i < ifade .length; i++){ //Örnek dizisinin her elemanı sırasıyla işleniyor.
        e = ifade [ i ];
        if (e. tip == 0){ //O andaki örnek bir işlenense
            c. cikinEkle (e ); //çıkına eklenmekte
        }else{ //eğer bir işlemse
            e2 = c. cikinSil (); //çıkından iki işlenen silip
            e1 = c. cikinSil (); //çıkından iki işlenen silip
            s = islem(e.islem, e1.islenen, e2.islenen ); //bu işlenenlerle işlem yapılmakta
            c. cikinEkle (s ); //ve bu işlemin sonucu yine çıkına eklenmekte
        }
    }
    e = c. cikinSil ();
    return e.islenen;
}
```

Arka gösterimi verilen bir ifadenin değerini ~ hesaplayan algoritma (2)

```
Ornek islem(char ch, int e1, int e2){
    int sonuc;
    switch (ch){
        case '+':sonuc = e1 + e2;
            break;
        case '-':sonuc = e1 - e2;
            break;
        case '*':sonuc = e1 * e2;
            break;
        case '/':sonuc = e1 / e2;
            break;
    }
    return new Ornek(sonuc);
}
```

Dört işlemi gerçekleştirmek için islem fonksiyonu kullanılmakta ve bu fonksiyon parametre olarak iki sayı ve yapılacak işlemi almaktadır.Fonksiyon parametre olarak aldığı işlemi yine parametre olarak aldığı iki sayı üzerinde gerçekleştirip,sonuç değerini bir örnek veri yapısı içinde döndürmektedir.

Soru : Arka gösterimi $ab/c - de + a$ olan bir ifadenin çıkın yardımıyla hesaplanması

\boxed{a} $b/c - de^* + a^*$	\boxed{b} a $/c - de^* + a^*$	$T_1 = a/b$ $\boxed{T_1}$ $c - de^* + a^*$	\boxed{c} $\boxed{T_1}$ $- de^* + a^*$
$T_2 = T_1 - c$ $\boxed{T_2}$ $de^* + a^*$	\boxed{d} $\boxed{T_2}$ $e^* + a^*$	\boxed{e} \boxed{d} $\boxed{T_2}$ $* + a^*$	$T_3 = d * e$ $\boxed{T_3}$ $\boxed{T_2}$ $+ a^*$
$T_4 = T_2 + T_3$ $\boxed{T_4}$ a^*	\boxed{a} $\boxed{T_4}$ $*$	$T_5 = a * T_4$ $\boxed{T_5}$	

1. a bir islenendir. Çıkına eklenir.
2. b bir islenendir. Çıkına eklenir.
3. / bir işlemidir. Çıkından iki eleman silinir (a ve b), bu iki elemanla / işlemi yapıp sonuç (T1) çıkına eklenir.
4. c bir islenendir. Çıkına eklenir.
5. - bir işlemidir. Çıkından iki eleman silinir (T1 ve c), bu iki elemanla - işlemi yapıp sonuç (T2) çıkına eklenir.
6. d bir islenendir. Çıkına eklenir.
7. e bir islenendir. Çıkına eklenir.
8. * bir işlemidir. Çıkından iki eleman silinir (d ve e), bu iki elemanla * işlemi yapıp sonuç (T3) çıkına eklenir.
9. + bir işlemidir. Çıkından iki eleman silinir (T2 ve T3), bu iki elemanla + işlemi yapıp sonuç (T4) çıkına eklenir.
10. a bir islenendir. Çıkına eklenir.
11. * bir işlemidir. Çıkından iki eleman silinir (a ve T4), bu iki elemanla * işlemi yapıp sonuç (T5) çıkına eklenir.

Ara Gösterimi Verilen Bir İfadenin Arka Gösterime Çevrilmesi

******Her iki gösterimde de işlenenlerin sırası aynı kalmakta , fakat işlemlerin yerleri değişmektedir.O halde bu defa işlenenleri değil işlemleri bir çıkına koyacak ve sırası gelen işlemi çıkından çıkarıp yazacağız.

Verilen bir ifadenin arka gösterimini bulan algoritma (2)

```

1 void arkaGosterim(Ornek[] ifade){
2 int i;
3 Ornek e, e1;
4 Cikin c = new Cikin(100);
5 for (i = 0; i < ifade .length; i++){
6 e = ifade [ i ];
7 if (e. tip == 0)
8 System.out.print(e.islenen );
9 else
10 if (e.islem == ' ( ' )
11 c. cikinEkle (e );

```

Algoritma girdi olarak bir dizi örnekten oluşan bir ifade almaktadır.Bu ifadenin her elemanı ya bir işlem yada işlenendir.Belirli bir anda ifadeden alınan eleman **e** olsun (6).Eğer **e** bir işlenense **e** nin çıkınla bir işi yoktur(7).Derhal ekrana yazılır(8).Eğer **e** bir işlemse , iki parantez karakterinden biri olabileceği gibi diğer işlemlerden biri de olabilir. Ch='>>Parantez kapama işlemi,önceliği en yüksek olan işlemidir(13).Çıkında o kapama parantezine karşılık gelen açma paratezi görülüne kadar(15) çıkındaki elemanlar çıkarılıp (17) ekrana yazdırılır(16). Ch='(' >> Parantez açma işlemi,önceliği en küçük olan işlemidir (10).Çıkının en üstüne eklenir(11). Ch=**İşlem** >> Eğer ch nın önceliği çıkının en üstündeki elemanın önceliğinden büyükse ch çıkına eklenir (26).Aksi takdirde çıkındaki elemanlar bu özellik sağlanana kadar çıkından çıkarılıp(23) ekrana yazılır(24) ve özellik sağlandığında ch çıkınına eklenir(26).

Bütün bu operasyonun sonunda eğer çıkında herhangi bir eleman kalmışsa(29) o eleman yada elemanlar çıkından birer birer çıkarılıp (30) ekrana yazdırılır(31).

```

12 else
13 if (e.islem == ' ') {
14 e1 = c. cikinSil ();
15 while (e1.islem != ' '){
16 System.out.print(e1.islem );
17 e1 = c. cikinSil ();
18 }
19 }
20 else{
21 while (e.oncelik <= c.ust (). oncelik ){
22 e1 = c. cikinSil ();
23 System.out.print(e1.islem );
24 }
25 c. cikinEkle (e );
26 }
27 }
28 while (!c.cikinBos ()){
29 e1 = c. cikinSil ();
30 System.out.print(e1.islem );
31 }
32 }

```

A	+	B	*	(
A	A	AB	AB	AB
(+	+	*	
*	*	*	+	
+	+	+		
C	+	D)	
ABC	ABC	ABCD	ABCD+	ABCD++

Ara gösterimi $A + B (C + D)$ olan bir ifadenin arka gösteriminin bir çıkın yardımıyla bulunması

1. A bir işlenendir. Ekrana yazılır.
2. + bir işlemidir. Çıkında eleman olmadığından çıkına eklenir.
3. B bir işlenendir. Ekrana yazılır.
4. * bir işlemidir. *'in önceligi çıkının en üstündeki +'dan büyük olduğundan çıkına eklenir.
5. (bir işlemidir. (doğrudan çıkına eklenir.
6. C bir işlenendir. Ekrana yazılır.
7. + bir işlemidir. +'nın önceligi çıkının en üstündeki ('den büyük olduğundan çıkına eklenir.
8. D bir işlenendir. Ekrana yazılır.
9.) bir işlemidir. Çıkından (görene kadar işlemler silinir ve ekrana yazılır.
10. Çıkında kalan * ve + sırasıyla çıkından silinip ekrana yazılır.

Infix'ten Prefix'e Dönüşüm

Sola ta ı ve parantez kaldır !

```

=((A+B) * (C+D))
= ((+AB)*(C+D))
= ( *(+AB)(C+D))
= (*(+AB)(+CD))
= (*+AB+CD)

```

Infix'ten Postfix'e Dönüşüm

Yı ıt kullanılır.Operatörler yı ıta yazılmaz.

1. Parantez açma
 2. Parantez kapama
 3. Durumlar
- Yı ıta “(“ görene dek tüm elemanlar çıkarılır.