Stack'ler Bilgisayar Dünyasında Nerelerde Kullanılır?

Yı ın mantı ı bilgisayar donanım ve yazılım uygulamalarında yaygın olarak kullanılmaktadır. Bunlardan bazıları;

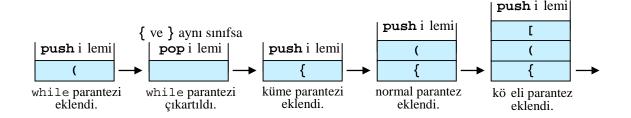
- Rekürsif olarak tanımlanan bir fonksiyon çalı ırken hafıza kullanımı bu yöntem ile ele alınır,
- (, {, [,], },) ayıraçlarının C/C++ derleyicisinin kontrollerinde,
- postfix → infix dönü türmelerinde,
- Yazılım uygulamalarındaki Parsing ve Undo i lemlerinde,
- Web browser'lardaki Back butonu (önceki sayfaya) uygulamasında,
- Ayrıca, mikroi lemcinin içyapısında **stack** adı verilen özel hafıza alanı ile mikroi lemci arasında, bazı program komutları ile (push ve pop gibi), bir takım i lemlerde (alt program ça ırma ve kesmeler gibi), veri transferi gerçekle ir. Mikroi lemcinin içinde, hafızadaki bu özel alanı gösteren bir yı ın i aretçisi (Stack Pointer –SP) bulunur. Stack Pointer o anda bellekte çalı ılan bölgenin adresini tutar. push fonksiyonu, stack'e veri göndermede (yazmada), pop fonksiyonu ise bu bölgeden veri almada (okumada) kullanılır.

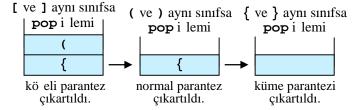
Bölüm 1'deki Özyinelemeli Fonksiyonlar konusunda gösterilen rekürsif faktöryel uygulamasında, fonksiyonun kendisini her ça ırmasında stack'e ekleme yapılır. stack'in en üstünde fact(0) bulunmaktadır. Fonksiyon ça ırıldı ı yere her geri döndü ünde ise stack'ten çıkarma i lemi yapılmaktadır.

Örnek 3.1: Anlamsız bir kod parçası olsun;

```
while(x == 7) {
    printf("%d", a[2]);
}
```

Üstteki kodlarda bulunan parantezlerin kullanımlarının do rulu u stack'lerle kontrol edilir. Bunun nasıl yapıldı ı alttaki ekilde adımlar halinde görülüyor. Bu uygulamada yukarıdaki kod parçasında bulunan bir parantez açıldı ında push() fonksiyonuyla stack'e ekleme, kapatıldı ında ise pop() fonksiyonuyla çıkarma yapaca ız.





pop() i leminde çıkarılacak parantez, kapanan parantezle uyu muyorsa hata durumu ortaya çıkacaktır. Son i lemden sonra stack'te eleman kalmadı ından, parantezlerin do ru bir ekilde kullanıldı ı görülmektedir.

ekil 3.8 Parantez kontrolünün ekilsel gösterimi.

3.4 INFIX, PREFIX VE POSTFIX NOTASYONLARI

Bilgisayarlarda infix yazım türünün çözümlenmesi zordur. Acaba x=a/b-c+d*e-a*c eklindeki bir ifadeyi çözümlerken, ((4/2)-2)+(3*3)-(4*2) gibi bir ifadenin de erini hesaplarken ya da a/(b-c)+d*(e-a)*c gibi parantezli bir ifadeyi i lerken derleyiciler sorunun üstesinden nasıl geliyor? 32*(55-32-(11-4)+(533-(533-(533+(533-(533+212)))*21-2))) gibi birçok operatör ve operand içeren bir i lemde nasıl operatör önceliklerine göre i lem sıralarını do ru belirleyip sonuç üretebiliyorlar?

Bir ifadede farklı önceliklere sahip operatörler yazılma sırasıyla i lenirse ifade yanlı sonuçlandırılabilir. Örne in 3+4*2 ifadesi 7*2=14 ile sonuçlandırılabilece i gibi 3+8=11 ile de sonuçlandırılabilir. Bilgisayarlarda infix yazım türünün çözümlenmesi zordur. Bu yüzden ifadelerin operatör önceliklerine göre ayrı tırılması, ayrılan parçaların sıralanması ve bu sıralamaya uyularak i lem yapılması gerekir. Bu i lemler için prefix ya da postfix notasyonu kullanılır. Ço u derleyici, kaynak kod içerisinde infix notasyonunu kullanmaktadır ve daha sonra stack veri yapısını kullanarak prefix veya postfix notasyonuna çevirir.

Bu bölümde bilgisayar alanındaki önemli konulardan biri olan infix, prefix ve postfix kavramları üzerinde duraca ız ve bu kavramlarda stack kullanımını görece iz.

- Operatörler : +, -, /, *, ^

- Operandlar : A, B, C... gibi isimler ya da sayılar.

Infix notasyonu: Alı a geldi imiz ifadeler infix eklindedir. Operatörlerin i lenecek operandlar arasına yerle tirildi i gösterim biçimidir. Bu gösterimde operatör önceliklerinin de i tirilebilmesi için parantez kullanılması arttır. Örne in infix notasyonundaki 2+4*6 ifadesi 2+24=26 ile sonuçlanır. Aynı ifadede + operatörüne öncelik verilmesi istenirse parantezler kullanılır; (2+4)*6. Böylece ifade 36 ile sonuçlandırılır.

Prefix notasyonu: Prefix notasyonunda (PN, *polish notation*) operatörler, operandlarından önce yazılır. Örne in 2+4*6 ifadesi infix notasyonundadır ve prefix notasyonunda +2*46 eklinde gösterilir. Benzer biçimde (2+4)*6 ifadesi *+246 eklinde gösterilir. Görüldü ü gibi prefix notasyonunda i lem önceliklerinin sa lanması için parantezlere ihtiyaç duyulmamaktadır.

Postfix notasyonu: Postfix notasyonunda (RPN, *reverse polish notation*) ise önce operandlar ve ardından operatör yerle tirilir. Aynı örnek üzerinden devam edersek; infix notasyonundaki 2+4*6 ifadesi prefix notasyonunda 2 4 6 * + eklinde, benzer biçimde (2+4)*6 ifadesi de 2 4 + 6 * eklinde gösterilir. Yine prefix'te oldu u gibi bu gösterimde de parantezlere ihtiyaç duyulmamaktadır. Bazı bilgisayarlar matematiksel ifadeleri postfix olarak daha iyi saklayabilmektedir.

Tüm aritmetik ifadeler bu gösterimlerden birini kullanarak yazılabilir. Ancak, bir yazmaç (*register*) yı ını ile birle tirilmi postfix gösterimi, aritmetik ifadelerin hesaplanmasında en verimli yoldur. Aslında bazı elektronik hesap makineleri (*Hewlett-Packard gibi*) kullanıcının ifadeleri postfix gösteriminde girmesini ister. Bu hesap makinelerinde biraz alı tırma yapıldı ında, iç içe birçok parantez içeren uzun ifadeleri, terimlerin nasıl gruplandı ını bile dü ünmeden, hızlı bir ekilde hesaplamak mümkündür.

lem önceli i;

- 1- Parantez içi
- 2- Üs alma
- 3- Çarpma/Bölme
- 4- Toplama Çıkarma

Aynı önceli e sahip i lemlerde sıra soldan sa a () do rudur. Yalnız üs almada sa dan sola do ru i lem yapılır.

Infix	Prefix	Postfix
A+B-C	-+ABC	AB+C-
(A+B)*(C-D)	*+AB-CD	AB+CD-*
A/B^C+D*E-A*C	-+/A^BC*DE*AC	ABC^/DE*+AC*-
(B^2-4*A*C)^(1/2)	(^-^B2**4AC/12)	(B2^4A*C*-12/^)
$A^B*C-D+E/F/(G+H)$	+-*ABCD//EF+GH	AB^C*D-EF/GH+/+
((A+B)*C-(D-E))^(F+G)	^-*+ABC-DE+FG	AB+C*DE-FG+^
A-B/(C*D^E)	-A/B*C^DE	ABCDE^*/-

Tablo 3.1 Matematiksel ifadelerde infix, prefix ve postfix notasyonunun gösterimi.

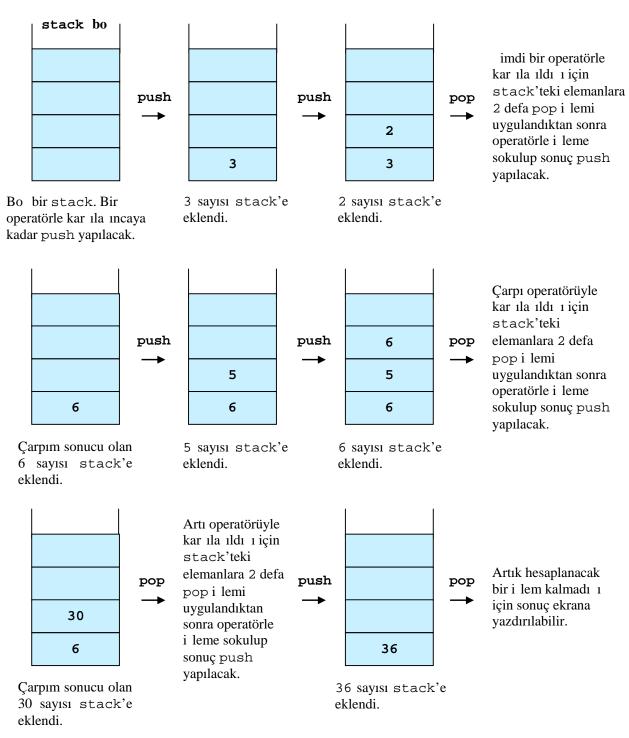
Dikkat edilecek olunursa, postfix ile prefix ifadeler birbirinin ayna görüntüsü de illerdir. imdi bir örnekle notasyon i lemlerinin stack kullanılarak nasıl gerçekle tirildiklerini görelim.

Örnek 3,2: 3 2 * 5 6 * + postfix ifadesini stack kullanarak hesaplayınız.

Çözüm algoritması öyle olmalıdır;

- 1- Bir operandla kar ıla ıldı ında o operand stack'e eklenir,
- 2- Bir operatör ile kar ıla ıldı ında ise o operatörün gerek duydu u sayıda operand stack'ten çıkarılır,
- 3- Daha sonra pop edilen iki operanda operatör uygulanır,
- 4- Bulunan sonuç tekrar stack'e eklenir.

Bu adımları uygulayarak örne in çözümünü ekillerle açıklayalım.



ekil 3.9 Postfix ifadesinin stack kullanılarak hesaplanması.