Veri Yapıları

YIĞIN / YIĞIT (Stack)

Dr. Sinan TUNCEL

Yığın (stack)

Dizinin herhangi bir elemanına, dizide bulunduğu adres kullanılarak doğrudan erişilebilir.
Örnek; dizinin 2. adresindeki eleman, yani a[2] ' nin değeri 4' dür.

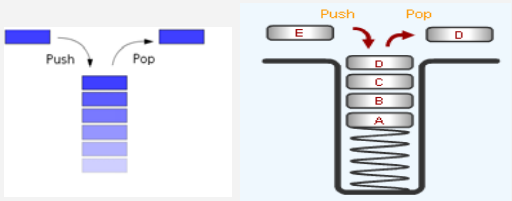
int a[5] = {3,5,4,1,7}

İndis	0	1	2	3	4
Değer	3	5	4	12	22

bazı uygulamalarda elemanlara belli bir sıraya göre erişmek gerekebilir

Yığın (stack)

Bilginin geliş sırasına göre, en son gelen elemana ilk erişilen liste yapısına **yığın (stack)** denir.
Bu yapıdaki listeler **LIFO-Last In First Out (Son Gelen İlk Çıkar)** listeler olarak da isimlendirilir.



Bir yığında bulunan herhangi bir objeye erişebilmek için, once bu objenin üstündeki diğer objelerin dışarı çıkması gerekir.

Yığın (Stack) Kulalnım alanları

- Örnek; üst üste konulan eşyaları taşımak için en üste konulan eşyayı (en son konulmuş olanı) ilk olarak almamız gerekir.
- Bir feribotun hem önünde hem arkasında araç indirme/bindirme kapısı varsa, o feribot FIFO düzeninde, sadece 1 kapısı varsa LIFO düzeninde araç indirip/bindirir.
- **Undo** işlemleri yığın ile yapılır.
- Web browser'larda önceki sayfaya ulaşmada(**Back)** yığın kullanır.
- Matematiksel işlemlerdeki operatörler (+,*,/,- gibi) ve operandlar için yığın kullanılabilir. (İŞLEM ÖNCELİĞİ)
- Yazım kontrolündeki parantezlerin kontrolünde yığın kullanılabilir

Yıgıtlar,
listelerin ters sırada yazdırılması, palindrom (okundugunda ve tersten okundugunda aynı sonucu veren karakter dizisi) benzeri yapıların bulunması,
bir ifadedeki parantez gibi sembollerin geçerliliginin test edilmesi, ifadelerin sonuçlarının hesaplanıp degerlerinin elde edilmesi,
infix ifadelerin postfix ifadeye dönüstürülmesi

Yığın (Stack) Kulalnım alanları

Fonksiyon ve Metot Çagrımlarında yığın kullanımı

Programlama dili derleyicileri (compiler) fonksiyon çagrımlarında da yıgıtlardan yararlanırlar.
Bir fonksiyon çagrıldıgında, çagıran fonksiyonun yerel degiskenleri sistem tarafından kaydedilmelidir; aksi halde çagrılan fonksiyon çagıran fonksiyonun degiskenlerini ve degerlerini ortadan kaldıracaktır.
Daha nalaşılır ifade edecek olursakyeni bir fonksiyonun çagrılması ile çagıran fonksiyonun degiskenleri ve geri dönülecek adres bir kagıda kaydedilir ve daha önce çagrılan fonksiyonlara iliskin bilgilerin tutuldugu kagıtların üzerine konulur. Bundan sonra denetim çagrılan fonksiyona geçer. Kendi degiskenlerine yer açarak onlar üzerinde islemler yapılmasını saglar.

Ayrıca çagıran fonksiyonda kalınan nokta (geri dönüs adresi), çagrılan fonksiyonun bitmesinden sonra geri dönmek üzere tutulmalıdır.

Yukarıdaki örnek ile ifade edecek olursak fonksiyondan geri dönülecegi zaman en üstteki kagıda bakılıp degiskenler üzerinde ilgili degisiklikler yapılarak geri dönüs adresine atlanır. Derleyici bu islemleri kagıt yerine yıgıt kullanarak gerçeklestirir.

Gerekli fonksiyonlar

Bu yapıyı gerçekleştirebilmek için gerekli fonksiyonlar şunlardır.
createStack : Belli miktarda eleman alabilecek boş bir yığın oluşturulur.
push : Verilen bir eleman tamamen dolu olmayan bir yığının en üstüne yerleştirilir. "push(s,i)", s yığınının en üstüne i değerini eleman olarak ekler.
pop :İçinde eleman olan bir yığının en üstündeki elemanı yığından alınır.
"i = pop(s)", s yığınının en üstündeki elemanı çıkartır ve değerini i değişkenine atar.
stackEmpty :Yığının boş olup olmadığı bilgisini verilir. empty(s), yığın bos ise TRUE değerini, değilse FALSE değerini döndürür
stackFull :Yığının dolu olup olmadığı bilgisini verilir.
stacktop: yığından çıkarılmaksızın en üstteki elemanın değerini döndürür

Not: Bos yığın üzerinden eleman çıkarılmaya veya yığının en üstündeki elemanın degeri belirlenmeye çalısıldıgında olusan geçersiz duruma **Underflow** . Bu nedenle Pop işlemi yapılmadan önce stackEmpty Push işlemi yapılmadan önce stackFull işlemi gerçekleştirilmelidir.

Yığın (stack)

Elimizde bir adet kitap (E) daha bulunsa idi Bu kitabı bu kutuya yerleştiremeyecektik.

Dolayısıyla yığınların kapasiteleri başlangıçta belirtilmelidir. (es:eleman sayısı)

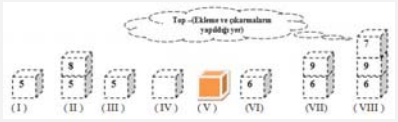
D
C
B
A

Şimdi yerleştirmiş olduğumuz bu kitaplardan C kitabını çıkarmak istesek, bunu tek seferde yapamayız. öncelikle D kitabını çıkarmamız gerekmektedir. yığından ilk önce yığına en son yerleştirilen eleman çıkacaktır.

Yığın örnek

Şekilde dizi boyutu 3 olan bir yıgın yapısında sırasıyla yapılan islemler gösterilmektedir.

(I) Yıgına '5' elemanı eklendi
(II) Yıgına '8' elemanı eklendi
(III) Yıgından eleman çıkarıldı
(IV) Yıgından eleman çıkarıldı
(V) Yıgından eleman çıkarıldı ("Yıgın bos" Hata mesajı verir)
(VI) Yıgına '6' elemanı eklendi
(VII) Yıgına '9' elemanı eklendi
(VIII) Yıgına '7' elemanı eklendi



Örnek

```
#include <iostream.h>
int top,maxlenght;
bool full() {
  if(top==maxlenght)
    return(true);
  else
    return(false);
}
bool empty(){
  if(top<=0)
    return(true);
  else
    return(false);
}
void POP(char &x,char S[]){
    if(empty())
      cout<<"yığın boş";
    else{
      x=S[top];
      top=top-1;
    }
}
```

```
void PUSH (char x,char S[]){
    if(full())
      cout<<"yığın dolu";
    else{
      top=top+1;
      S[top]=x;
    }
}
main(){
    int i;
    char ch,Yigin[50];
    char str[50];
    cout<<"bir metin giriniz";
    gets(str);
    top=0;
    maxlenght=sizeof(Yigin);
  for(i=0;i<strlen(str);i++)
    PUSH(str[i],Yigin);
  while(empty()==false){
      POP(ch,Yigin);
      cout<<ch;
  }
system("pause");
}
```


Örnek Soru

1-Verilen bir cümlenin palindorme (düzden ve tersten aynı okunan sözcükler) olup olmadığını belirleyen algoritmayı oluşturunuz.

Algoritma:

İlk olarak cümledeki boşlukları silin.

Daha sonra tüm cümleyi bir yığına yerleştirin.

Daha sonra yığından tek tek çekme (pop) yapın boşlukları kaldırılmış cümlenin karakterleri ile çektiğiniz karakterleri birer birer karşılaştırın.

2- Şekilde gördüğünüz diskleri ilk çubuktan üçüncü çubuğa yine aynı sıra ile (aşağıdan yukarı büyükten küçüğe) taşıyın.

Bir defada sadece 1 disk alıp başka bir çubuğa yerleştirebilirsiniz.

Küçük diskin üzerine büyük disk yerleştiremezsiniz.

Yığın kullanımı (parantez kontrolü)

İç içe parantezler içeren bir ifadede parantezlerde hata kontrolünün yapısı ile gerçekleştirilmesi;

1. Açılan ve kapanan toplam parantez sayısı eşit olmalıdır.
2. 2. Kapanan her parantezden önce bir parantez açılmış olmalıdır.

"((A+B)" ve "A+B(" 1. sarta uymaz.

")A+B(-C" ve "(A+B))-(C+D" 2. sarta uymaz.

İfadenin herhangi bir noktasındaki o ana kadar açılmıs fakat kapanmamıs parantezlerin sayısı Nesting Depth (parantez derinligi) dir

Yığın kullanımı (parantez kontrolü)

Parantezleri geçerli bir ifadede su şartlar olmalıdır

1. İfadenin sonunda parantez sayısı 0 olmalıdır. İfadede ya hiç parantez yoktur veya açılan parantezlerin sayısı ile kapanan parantezlerin sayısı eşittir.
2. İfadenin hiçbir noktasında parantez sayısı negatif olmamalıdır. Bu, parantez açılmadan bir parantezin kapanmadığını garantiler.

Yandaki ifade iki şarta da uyar:

7- ((x *((x+y) / (j-3))+y) /(4 -2.5))
0012223444433444432221122 2 2 10 (iki şartada uyar)

((A+B)
122221 (sonunda 1 parantez arttı.)

(A+B)) - (C + D
1111 0-1-10 0 0 0 (Arada negatif oldu)

Yığın kullanarak parantez kontrol:

1) Boş bir yığın oluştur ve sembolleri okumaya başla
2) Eğer sembol başlangıç sembolü ise ('(','[','{') Yığına koy
3) Eğer sembol kapanış sembolü ise (')',']','}')
   I. Eğer yığın boşsa hata raporu döndür
   II. Değilse
       Yığından al
       Eğer alınan sembol ile başlangıç sembolü aynı
       değilse hata gönder
4) İfade bitti ve yığın dolu ise hata döndür.

13

Yığın kullanımı (parantez kontrolü)

Problem parantezler "(", ")" , köseli parantezler "[", "]"

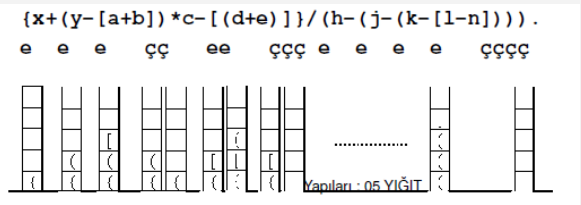
Küme parantezleri : "{", "}" içeriyorsa parantez dogrulugu kontrolü için parantez sayılarının yanında tiplerinin de tutulması gerekir. Bu nedenle yukarıdaki yöntemin kullanılması uygun olmaz.

void main() { printf("Merhaba"};); (yanlıs bir ifadedir.)

14

Yığın kullanımı (parantez kontrolü)

Karsılasılan parantezleri tutmak üzere yığın kullanılabilir, Bir parantezle karsılasıldıgında yığına eklenir. İgili parantezlerin karsılıgı ile karsılasıldıgında ise yığına bakılır. yığın bos degilse yığından bir eleman çıkarılarak dogru karsılık olup olmadıgı kontrol edilir. Dogruysa islem sürdürülür. Degilse ifade geçersizdir. yığın sonuna ulasıldıgında yığın bos olmalıdır. Aksi halde açılmıs ama kapanmamıs parantez olabilir



15

infix, postfix, prefix

Bir programlama dilinde, bir aritmetik işlem, yazılış sırasına göre değil, işlem öncelikleri sırasına göre yapılır

A + B * C / D işleminde * ve / işlemleri + işleminden öncelikli oldukları için işlem sırası :
1. B' yi C ile çarp
2. Sonucu D' ye böl
3. Bölme işleminden elde ettiğin sonucu A' ya ekle

olmalıdır.

in, pre ve post, operatör'ün operand'lara göre yerine karsılık gelir.

A+B işleminde
operatör (islemci) : +
operands (islenenler) : A, B
infix gösterim : A+B
prefix gösterim : +AB (benzeri bir gösterim add(A,B) fonksiyonu)
postfix gösterim : AB+

16

infix, postfix, prefix

A+B*C infix ifadesini postfix'e çevirelim.
işlem önceliğine göre çarpmanın toplamaya önceliği olduğu için, A+(B*C) seklinde düşünülebilir.
Önce çarpma kısmı postfix'e çevrilecek sonra da sonucu.

A+(B*C) anlaşılırlığı artırmak için parantez kullandık.
A+(BC*) çarpım çevrildi.
A(BC*)+ toplam çevrildi.

İslem önceligi (büyükten küçüge)
Üs alma
Çarpma/Bölme
Toplama/Çıkarma

Parantezsiz ve aynı öncelige sahip islemcilerde islemler soldan saga dogru yapılır (üs alma hariç). Üs almada sagdan sola doğrudur. A-B+C 'de öncelik (AB)+C seklindedir. A^B^C'de ise A^(B^C) seklindedir. Parantezler default öncelikleri belirtmek için konulmustur.

17

Postfix Formun Özellikleri

Postfix formda parantez kullanımına gerek yoktur.
A+(B*C) ve (A+B)*C
Yukarıda infix ifadede ilk ifadede parantez gereksizdir. kincide ilk ifade ile karıştırılmaması için gereklidir. Postfix forma çevirmek bu karışıklığı önler. (A+B)*(C+D)'yi infix formda parantezsiz ifade etmek için çarpım işlemi yapılırsa işlem sayısı çoğalır.

Infix formdan postfix forma çevrilen bir ifadede operand'ların (sayı veya sembol) bağlı olduğu operatör'leri (+,-,*,/) görmek zorlaşır (3 4 5 * + ifadesinin sonucunun 23'e, 3 4 + 5 * ifadesinin sonucunun 35'e karşılık geldiğini bulmak zor gibi görünür). Fakat parantez kullanmadan tek anlama gelen bir hale dönüşür. işlemleri, hesaplamaları yapmak kolaylaşır.

18

infix ifadenin postfix ifadeye cevrilmesi

Infix formunda yazılmış bir notasyonun postfix notasyonuna çevrilmesinde yığın kullanılır.

İşlem işaretleri postfix notasyonuna çevrilecek aritmetik işlemde yazılış sırasına göre, kendisinden sonra gelen **kendisiyle aynı veya düşük öncelikli** ilk işlem işaretine kadar yığında bekletilir.

19

infix ifadenin postfix ifadeye cevrilmesi

A + B * C / D + E ifadesinin postfix notasyonuna dönüştürülmesini gösteren animasyon:

A + B * C / D + E

20

Çevrim Örnekleri

infix	posfix	Prefix
A+B-C	AB+C-	-+ABC
(A+B)*(C-D)	AB+CD-*	*+AB-CD
A^B*C-D+E/F/(G+H)	AB^C*D-EF/GH+/+	+-*ÂBCD//EF+GH
((A+B)*C-(D-E))^(F+G)	AB+C*DE—FG+^	^-*+ABC-DE+FG
A-B/(C*D^E)	ABCDE^*/-	-A/B*C^DE

21

Çevrim Örnekleri

infix	posfix
A+B-C	AB+C-
(A+B)*(C-D)	AB+CD-*
A^B*C-D+E/F/(G+H)	AB^C*D-EF/GH+/+
((A+B)*C-(D-E))^(F+G)	AB+C*DE—FG+^
A-B/(C*D^E)	ABCDE^*/-
((A / (B ^ C)) – ((D * E) – (A * C)))	

22

Postfix ifadenin Sonucunun Hesaplanması

Postfix notasyonuna çevrilmiş bir aritmetik işlemin sonucunun elde edilmesi için de yığın kullanılır.

İşleme girecek elemanlar herhangi bir işlem işareti gelene kadar yığına yerleştirilir.

İşlem işareti geldiğinde yığının üstündeki iki eleman yığından çekilerek aralarında işlem yapılır. İşlem sonucu tekrar yığına yerleştirilir.

Bir postfix string'inde her operatör kendinden önce gelen iki operand üzerinde islem yapacaktır.

Bu operandlar daha önceki operatör'lerin sonuçları da olabilir.

Bir anda ifadeden bir operand okunarak yığına yerleştirilir.

Bir operatör'e ulaşıldığında, yığının en üstündeki iki eleman bu operatör'ün operand'ları olacaktır. İki eleman yığından çıkarılır işlem yapılır ve sonuç tekrar yığına konulur.

Artık bir sonraki operatör'ün operand'ı olmaya hazırdır.

Birçok derleyici 3*2+5*6 gibi bir infix ifadenin değerini hesaplayacağı zaman postfix forma dönüştürdükten (belirsizliği ortadan kaldırdıktan sonra) sonucu hesaplar :

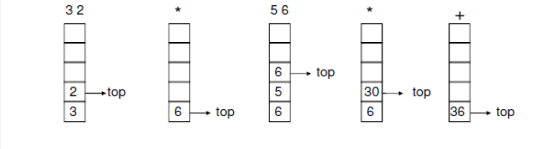
"3 2 * 5 6 * +"

23

postfix ifadenin sonucunun hesaplanması

Bir postfix ifadenin "3 2 * 5 6 * +" sonucunun hesaplanması

Bir operatöre ulaşıldığında, son iki eleman yığından çıkarılır, islem yapılır, sonuç yığına yerleştirilir:



24

postfix ifadenin sonucunun hesaplanması

ABC*D/+E+ postfix notasyonunun sonucunun hesaplanmasını gösteren animasyon:

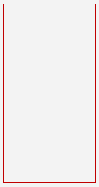
9 8 8 6 - / 2 * 1 + /

8 9

1

2

4



25

postfix ifadenin sonucunun hesaplanması

Örnek: 9 8 8 6 - / 2*1+/ = ?

Okunan Simge	yığın	İşlem
9	9	-
8	9 8	-
8	9 8 8	-
6	9 8 8 6	-
-	9 8 2	8-6=2
/	9 4	8/2=4
2	9 4 2	-
*	9 8	4*2=8
1	9 8 1	-
+	9 9	8+1=9
/	1	9/9

26

İyi çalışmalar…

27