YIĞIN YAPISI

Eleman ekleme çıkarmaların en üstten (top) yapıldığı veri yapısına yığın (stack) adı verilir. Bir eleman ekleneceğinde yığının en üstüne konulur. Bir eleman çıkarılacağı zaman yığının en üstündeki eleman çıkarılır. Bu eleman da yığındaki elemanlar içindeki en son eklenen elemandır. Bu nedenle yığınlara LIFO (Last In First Out: Son giren ilk çıkar) listesi de denilir.

4
3
2
1

Yığın tanımları:

Boş yığın (empty stack): Elemanı olmayan yığıt.

push (yığına eleman ekleme): "push(s,i)", s yığınının en üstüne i değerini ekler.

pop (yığından eleman çıkarma): "i = pop(s)", s yığıtının en üstündeki elemanı çıkartır ve değerini i değişkenine atar.

empty (yığının boş olup olmadığını belirleyen işlem): empty(s), yığın boş ise TRUE değerini, değilse FALSE değerini döndürür.

Underflow: Boş yığın üzerinden eleman çıkarılmaya veya yığının en üstündeki elemanın değeri belirlenmeye çalışıldığında oluşan geçersiz durum.

Overflow: Dolu yığına eleman eklenmeye çalışıldığında oluşan durum.

Dizi ile Yığın Örneği:

```
#define N 100
int Yigin[N],indis=0;
int Yigina_Ekle(int veri)
{
        if (indis>=N)
        {
                puts("Yigin Dolu"); return -1;
        }
        else
        {
                Yigin[indis]=veri; indis++;
        }
}
int Yigindan_Al()
{
        if (indis<=0)
        {
                puts("Yigin Bos");return -1;
        }
        else
                return Yigin[--indis];
 }
```

Örnek Yiğin:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
typedef struct lnode{
        int data;
        struct lnode * next;
}StackList;
StackList *iter=NULL;
void push(int data){
    if(iter==NULL){  //stack bos ise
       iter = (StackList * ) malloc(sizeof(StackList));
        iter->next=NULL;
        iter->data = data;
    }
    else{            //stack bos degil ise
       StackList * yeni = (StackList *)malloc(sizeof(StackList));
       yeni->next = iter;
       yeni->data=data;
       iter=yeni;
       }
}
int pop(){
       if(iter == NULL)
              return -1;
       int result = iter->data;
       StackList *temp = iter;
       iter=iter->next;
       free(temp);
       return result;
}
void printStack(){
    printf("\n");
    StackList *temp = iter;
    while(temp!=NULL){
        printf("%d _ ",temp->data);
        temp=temp->next;
    }
}
int main(){
       push(10);
       printStack();
       push(20);
       printStack();
       push(30);
       printStack();
       printf("\npop> %d",pop());
       printStack();
       printf("\npop> %d",pop());
       printStack();
       printStack();
       printf("\npop> %d",pop());
       printStack();
       printf("\npop> %d",pop());
       printStack();
       getch();
       return 0;
}
```

Yığın ile Desimal den Binary'e dönüşüm:

```
void cevir(int sayi){
        int digit;
        while(sayi>0)
        {
                digit=sayi%2;
                push(digit);
                sayi=sayi/2;
        }
        while (2>1)
        {
                digit=pop();
                if (digit>-1)
                        printf("%d",digit);
                else
                        break;
        }
}
```

```
int main(){
        int numara;
        system("cls");
        puts("\n 2lige cevrilecek Sayi?");
        scanf("%d",&numara);
        cevir(numara);
        return 0;
```

Yığın ile Parantez Eşleme Örneği:

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.

```
int Parantez_Kontrol(char *islem)
{
   int i;
        for (i=0;i<strlen(islem);i++)
        {
                if (islem[i]=='(')
                        push(islem[i]);
                else
                {
                        if (islem[i]==')')
                                if(pop()==-
1)

        return -1;
                }
        }
        if (pop()!=-1)
                return 0;
        else
                return 1;
}
```

```
}

int main()
{
        int i;
        char secim;
        char islem[100];//="(3+4)/5)";
        printf("Kontrol stringi=>");
        scanf("%s",islem);
        i=Parantez_Kontrol(islem);
        switch(i)
        {
        case -1:printf("Acma Parantezi Eksik");
                        break;
        case 0:printf("Kapama parantezi Eksik");
                break;
        case 1:printf("Parantez Hatasi Yok");
        }
        return 0;
```

Infix den Postfix'e Dönüşüm:

Infıx: a+b : operatör operandların arasında.
Postfix: ab+ : operator operantlardan sonra.
Prefix: +ab : operator operantlardan önce.

Infix notasyonun dezavantajı operatörlerin değerlendirme kontrolu için parantezler kullanılmasıdır. Postfix ve prefix notasyonlarda parantezler kullanılmaz. Infix notasyonu kullanan yüksek seviyeli dillerde ifadeler direkt değerlendirilemezler. Infix notasyondaki ifadeler postfix notasyona çevrilerek değerlendirilir.

Gönderilen > yığın içindeki ise yığına eklenir <= ise yığından operatör çıkarılır.

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

Örnek: A+B*C-D/E

Infix	Yığın	Postfix
A+B*C-D/E		
+ B*C-D/E		A
B*C-D/E	+	A
*C-D/E	+	AB
C-D/E	* +	AB
-D/E	* +	ABC
-D/E	+	ABC*
-D/E		ABC*+
D/E	-	ABC*+
/E	-	ABC*+D
E	/ -	ABC*+D
	/ -	ABC/+DE
	-	ABC/+DE/
		ABC/+DE/-

Örnek: A*(B+C)*D

Infix	Yığın	Postfix
A*(B+C)*D		
*(B+C)*D		A
(B+C)*D	*	A
B+C)*D	(*	A
+C)*D	(*	AB
C)*D	+ (*	AB
)*D	+ (*	ABC
*D	(*	ABC+
*D	*	ABC+
*D		ABC+*
D	*	ABC+*
	*	ABC+*D
		ABC+*D*

Postfix ifadelerin değerlendirilmesi: Eğer sayı gelirse yığına atılır. Eğer matematiksel operatör gelirse iki sayı yışından çıkarılır işlem yapılır ve sonuç yığına atılır.

Postfix	Yığın
1 2 3 + *	
2 3 + *	1
3 + *	2 1
+ *	3 2 1
*	(2+3=5) 1
	(1*5=5)

Postfix değerlendirme algoritması:

```
Algoritma posfix_degerlendir (değer ifade <string>)
İfade_boyutu=string boyutu
Yeni_Yığın=Yığın_Oluştur();
İndex=0;
Loop (index<İfade_Boyutu)
        if (ifade[index] operand)
                Yigina_Ekle(Yeni_Yığın,ifade[index]);
        Else
                operand2=Yigindan_Cikar(Yeni_Yığın);
                operand1=Yigindan_Cikar(Yeni_Yığın);
                operator=ifade[index];
                değer=hesapla(operand1,operator,operand2);
                Yığına_Ekle(Yeni_Yığın,değer);
        End if;
        index=index+1;
end loop;
        sonuç=Yığından_Çıkar(Yeni_Yığın);
        return sonuç;
end Algoritma;
```