Veri Yapıları
Lab Notları - 3

Özyineleme (Recursion)

Bir fonksiyon veya metodun kendi kendini çağırma işlemine özyineleme denir. Özyinelemenin 2 temel kuralı vardır.
- Sonlanacağı durum (Temel adım)
- Her çağrıda sonlanacağı duruma yaklaşması (Özyineleme)

Eğer bir sonlanma durumu olmaz ise sonsuz çağrım olacak ve program hata verecektir. Sonlanma durumu var ama her çağrıda o duruma yaklaşılmaz ise yine sonsuz çağrım olur. Özyineleme birçok problemde kod satırlarının, bir iki satıra inmesini sağlar. Fakat hız performansı olarak sürekli iç içe çağrımlar yapıldığı için bellekte sürekli bir yer ayırma olacaktır bundan dolayı hesaplama hızı düşecektir.

Toplam: 1'den girilen sayıya kadar bütün tam sayıların toplamını döndüren özyinelemeli fonksiyon

```
int Toplam(unsigned int sayi){
	if(sayi == 0) return 0;
	return sayi + Toplam(sayi-1);
}
```

Faktöriyel: Girilen sayının özyineleme kullanarak faktöriyelini alan fonksiyon

```
int Fakt(unsigned int sayi){
	if(sayi == 0) return 1;
	return sayi * Fakt(sayi-1);
}
```

Fibonacci: Fibonacci hesabını özyinelemeli olarak yapan fonksiyon

```
int Fib(unsigned int sayi){
	if(sayi == 1 || sayi == 2) return 1;
	return Fib(sayi-1) + Fib(sayi-2);
}
```

Asal Kontrolü: Burada C++'ta desteklenen varsayılan parametre özelliği kullanılarak hesaplama yapılmıştır. İkinci parametre girilmediği sürece değeri varsayılan olarak ikidir.

```
bool Asal(int sayi, int bolen=2){
	if(sayi == bolen || sayi == 1) return true;
	if(sayi % bolen == 0) return false;
	return Asal(sayi,bolen+1);
}
```

1'den girilen sayıya kadar ekrana yazılması

```
void Yaz(int sayi){
	if(sayi==0)return;
	Yaz(sayi-1);
	cout<<sayi<<" ";
```

```
}
```

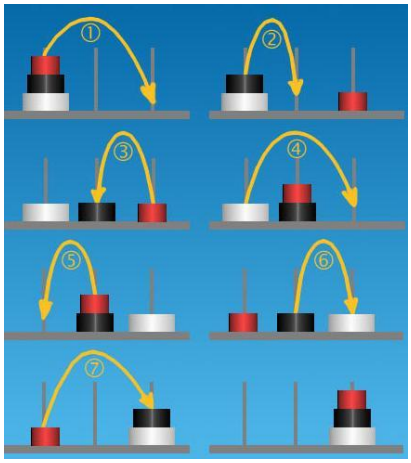
Üs: Bir sayının üssünün özyinelemeli olarak alınması

```
int Us(int sayi,int us)
{
  if(us==0)return 1;
  return sayi*Us(sayi,us-1);
}
```

String uzunluğu: Bir string'in uzunluğunu özyinelemeli olarak bulan fonksiyon

```
int Uzunluk(string yazi){
        if(yazi[0] == '\0') return 0;
        return 1 + Uzunluk(yazi.substr(1));
}
```

Hanoi Kuleleri Çözümü

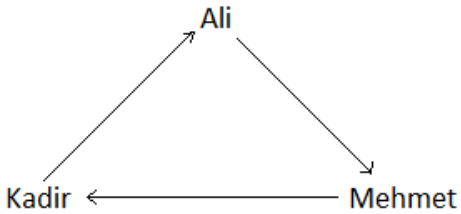



```
#include <iostream>
using namespace std;
void Hanoi(int diskSayisi, char kaynakKule, char hedefKule, char geciciKule){
        if(diskSayisi > 1) {      // geçici kuleye ihtiyaç var
                Hanoi(diskSayisi-1,kaynakKule, geciciKule, hedefKule);
                Hanoi(1,kaynakKule, hedefKule, geciciKule);
                Hanoi(diskSayisi-1,geciciKule, hedefKule, kaynakKule);
        }
        else {    // Tek hamlelik durum
                cout<<"Bir diski "<<kaynakKule<<" den "<<hedefKule<<" ye tasi"<<endl;
        }
}
int main(){
        int diskSayisi;
        cout<<"Disk Sayisi:";
        cin>>diskSayisi;
        Hanoi(diskSayisi, 'A', 'B', 'C');
        return 0;
}
```

Buraya kadar gösterilen örnekler özyinelemenin direk olarak kullanılmasıydı. Fakat aşağıda verilen örnekte özyinelemenin dolaylı olarak kullanılması söz konusudur.

```
#include <iostream>
using namespace std;
void Ali(int); // prototip tanımı
void Kadir(int devam){
        if(!devam) return;
        cout<<"Kadir"<<endl<<"----------"<<endl;
        Ali(devam-1);
}
void Mehmet(int devam){
        if(!devam) return;
        cout<<"Mehmet"<<endl;
        Kadir(devam);
}
void Ali(int devam){
        if(!devam) return;
        cout<<"Ali"<<endl;
        Mehmet(devam);
}
int main(){
        int sayi;
        cout<<"Kac Kere Cagrilsin:";
        cin>>sayi;
        Ali(sayi);
   return 0;
}
```

Yukarıdaki kod incelendiğinde Ali metodu, Mehmet metodunu çağırıyor, Mehmet metodu da Kadir metodunu çağırıyor, Kadir metodu tekrar Ali metodunu çağırıyor. Görülüyor ki dolaylı bir şekilde özyineleme çağrımı yapılmıştır. If kontrolü içerisinde tam sayının yazılması daha öncede bahsedildiği üzere C++'ta sıfır haricindeki bütün sayılar doğru olarak işlem görmektedir. Dolayısıyla biz main içerisinde Ali metoduna kaç kere çağrıldığını parametre olarak verip Kadir metodunun içerisinde de bu değeri bir azaltıyoruz.



Hazırlayan
Arş. Gör. Dr. M. Fatih ADAK