**Algoritma ve Programlama – I**

Proje 2   
Kapalı Sayılar ve Seriler

05100000003 – Abdulbaki Furkan TANRIVERDİ  
05110000064 – Onur Cem ŞENEL

**Teslim Tarihi:** 15.12.2011

**İçindekiler**

**1.** Kaprekar Sabitinin Tanımı…………………………………………………………….3

**2.** Analiz…………………………………………………………………………………………...3

**3.** Tasarım………………………………………………………………………………………..4  
 **3.1.** Programda Kullanılan Fonksiyonlar…………………………………………4  
 **3.2.** Programda Kullanılan Diziler…………………………………………………..5  
 **3.3.** Programın Algoritması…………………………………………………………….5

**4.** Programcı Kataloğu……………………………………………………………………...8

**5.** Kullanıcı Kataloğu……………………………………………………………………….12  
 **5.1.** Programdaki Kısıtlamalar………………………………………………………12

**1. Kaprekar Sabitinin Tanımı**

Kapalı sayı kavramı ilk defa 2004 yılında Dj.A.Babayev tarafından tanımlamıştır. Bu kavram 1946 yılında Hintli matematikçi D.R. Kaprekar tarafından bulunan Kaprekar sabitinin (6174) farklı basamak sayıları ve farklı tabanlar için genelleştirilmesidir. N, dört basamaklı, tüm rakamları aynı olmayan bir tamsayı olsun. N tamsayısının rakamları artan ve azalan şekilde yeniden sıralandığında, küçük sayıya S, büyük sayıya L denir. Büyük sayıdan küçük sayı çıkarılır ve sonuca R (R=L-S) denir. Çıkan sonuç için aynı sıralama ve çıkarma işlemi devam ettirildiğinde, kendini tekrar eden sabit bir sayıya (6174) ulaşılır.

Verilen N tamsayısı için sıralama ve çıkarma işlemleri sonucunda iki durum ortaya çıkar. Birinci durumda, N tamsayısı sıralama ve çıkarma işlemleri sonunda kendini tekrar eden bir dizi (seri) ile sonlanır. İkinci durumda, N tamsayısı sıralama ve çıkarma işlemleri sonunda yukarıda da belirtildiği gibi sabit bir sayı (kapalı sayı), yani uzunluğu 1 olan seri, ile sonlanır.

Eğer verilen N tamsayısında sıfırlar var ise, S sayısının başında bu sıfırlar korunur. Örneğin; N=21003 için, L=32100 ve S=00123’tür. Ayrıca sıralama ve çıkarma işlemleri sırasında, verilen N tamsayısından daha az basamak sayısına sahip sayılarla karşılaşılabilir. Yani, kalan R sayısı, N tamsayısından daha az basamak sayısına sahip ise L sayısının sonuna ve S sayısının başına gerektiği kadar 0 eklenir. Örneğin; N=2122 için L=2221, S=1222 ve R=999 olur. Bu durumda bir sonraki işlem için L=9990, S=0999 şeklinde olacaktır. Böylece, sıralama ve çıkarma işlemleri sırasında tüm sayıların aynı sayıda basamağa sahip olması sağlanır.

**2. Analiz**

Bizim programımız, kullanıcıdan 2 ve 9 basamaklı sayıları içeren bir sayı alıyor. Daha sonra alt programlar ve döngüler sayesinde seri ya da kapalı sayıya ulaşıldığı, seri elemanları, kapalı sayı ise bu sayının değeri ve kaç adımda bulunduğu yazdırılacak.

Programımızda girilen sayının 2 ve 9 basamaklı sayılara dahil olduğu kabul edilip, daha az veya çok basamak girilmediği varsayılıyor. Sadece sayının bütün basamaklarının aynı olup olmamasını “do-while” ile kontrol edeceğiz. Ayrıca yapılan çıkarma işlemlerinden sonra eğer oluşan yeni sayı basamak sayısı olarak 1 azalırsa, 10 ile çarpacağız. Dikkat edilmesi gereken noktalar bunlar.

Bu programımızın amacı tek boyutlu dizilerinin kullanımı ve “arama”-“sıralama” fonksiyonlarının etkin olarak kullanmaktır.

**3. Tasarım**  
 **3.1. Programda Kullanılan Fonksiyonlar**

**•void yerdegistir(int \*,int \*);**

Bu fonksiyonu “dizi\_sirala\_min\_max “ ve “dizi\_sirala\_max\_min” fonksiyonlarının içinde kullandık. Amacımız, girilen sayıların basamaklarına ayırmak ve sayıları yeniden sıralamak. Sayılar arasındaki yer değiştirme için de bu fonksiyon kullanıldı. Parametreler ise işaretçi tipinde. Çünkü sayıların değerlerini alıyor ve onları değiştiriyor. Bu nedenle geriye bir sayı döndürmüyor.

**•void dizi\_sirala\_min\_max(int \*,int);**

Fonksiyonumuz dizinin elemanlarını ve girilen sayının basamak sayısını alıp, “iç içe for” ile bu sayılar küçükten büyüğe sıralıyor. Ayrıca dizimiz “pointer” ile tanımlandığından geriye bir değer döndürmüyor. Bu fonksiyonun içinde sayaç olarak kullandığımız ve “int” tipinde “i” ve “j”; ve küçük sayıyı tutması için “min\_index” tanımladık.

**•void dizi\_sirala\_max\_min(int \*,int);**

Fonksiyonumuz dizinin elemanlarını ve girilen sayının basamak sayısını alıp, “iç içe for” ile bu sayılar büyükten küçüğe sıralıyor. Ayrıca dizimiz “pointer” ile tanımlandığından geriye bir değer döndürmüyor. Bu fonksiyonun içinde sayaç olarak kullandığımız ve “int” tipinde “i” ve “j”; ve küçük sayıyı tutması için “max\_index” tanımladık.

**•int dizide\_ara(int \*,int,int);**

Bu fonksiyona diziyi, seri ya da kapalı olup olmadığını anlamak ve kontrol etmek için elde ettiğimiz sayıyı ve belli sayıda döndürmek için sayacı gönderiyoruz. Bu fonksiyonda da “for” döngüsünü saydırmak için “int” tipinde “i” ve “kontrol” değişkenini atadık. Çıkan sonuca göre “kontrol” değişkeni geriye farklı değerler döndürücek.

**•int diziden\_sayi\_hesapla(int \*,int);**

Bu fonksiyonumuz, diziye atadığımız basamak değerlerini tekrardan bir araya getirip, yeni sayıyı oluşturuyor. “long int” tipinde “sonuç” olarak tanımladığımız değişken ile çağırılıyor.

**•int bas\_say\_bul(long int);**

Basamak sayısını bulmak için kullandığımız bu fonksiyon “do-while” döngüsü yardımıyla kontrol ediliyor. Ve basmak sayısını geri döndürüyor.

**•long int us\_al(int,int);**

Üs alma işine yarayan bu fonksiyon, sayıdaki değerleri diziye atamak ve tam tersi işlemleri yapmak için kullanılıyor. Tanımlanan değişkenler taban ve üstür.

**3.2. Programda Kullanılan Diziler**

**•int sayi\_dizi[BOYUT\_SAYI]={0};**

Girilen sayının basamaklarını tutmak için int tipindeki matris.

**•int fark[BOYUT\_FARK]={0};**

Sayılar küçükten büyüğe ve büyükten küçüğe tanımlanıp bunların farkı alındığında oluşan yeni sayıyı tutan dizi.

**3.3. Programın Algoritması**

|  |
| --- |
| **D:\Resimler\MP Navigator EX\2011_12_13\IMG_0003.jpg**  **D:\Resimler\MP Navigator EX\2011_12_13\IMG_0002.jpg** |

|  |
| --- |
| **D:\Resimler\MP Navigator EX\2011_12_13\IMG_0001.jpg** |

**4. Programcı Kataloğu**

Bu programda analiz, tasarım, gerçekleştirim, test ve raporlama için yaklaşık 10 saat harcandı.

**Kaynak Kod**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define BOYUT\_SAYI 9**

**#define BOYUT\_FARK 100**

**void yerdegistir(int \*,int \*);**

**void dizi\_sirala\_min\_max(int \*,int);**

**void dizi\_sirala\_max\_min(int \*,int);**

**int dizide\_ara(int \*,int,int);**

**int diziden\_sayi\_hesapla(int \*,int);**

**int bas\_say\_bul(long int);**

**long int us\_al(int,int);**

**int main()**

**{**

**int i,sayac=0,kontrol=0,kucuk\_sayi,buyuk\_sayi;**

**int sayi\_dizi[BOYUT\_SAYI]={0};**

**int fark[BOYUT\_FARK]={0};**

**long int sayi;**

**do{**

**printf("En az 2, en fazla 9 basamakli pozitif bir tam sayi giriniz: ");**

**scanf("%ld",&sayi);**

**for(i=0;i<BOYUT\_SAYI;i++)**

**{**

**sayi\_dizi[i]=sayi%us\_al(10,i+1)/us\_al(10,i);**

**}**

**dizi\_sirala\_min\_max(sayi\_dizi,bas\_say\_bul(sayi));**

**buyuk\_sayi=diziden\_sayi\_hesapla(sayi\_dizi,bas\_say\_bul(sayi));**

**dizi\_sirala\_max\_min(sayi\_dizi,bas\_say\_bul(sayi));**

**kucuk\_sayi=diziden\_sayi\_hesapla(sayi\_dizi,bas\_say\_bul(sayi));**

**}while(kucuk\_sayi==buyuk\_sayi);**

**do{**

**for(i=0;i<BOYUT\_SAYI;i++)**

**{**

**sayi\_dizi[i]=sayi%us\_al(10,i+1)/us\_al(10,i);**

**}**

**dizi\_sirala\_min\_max(sayi\_dizi,bas\_say\_bul(sayi));**

**buyuk\_sayi=diziden\_sayi\_hesapla(sayi\_dizi,bas\_say\_bul(sayi));**

**dizi\_sirala\_max\_min(sayi\_dizi,bas\_say\_bul(sayi));**

**kucuk\_sayi=diziden\_sayi\_hesapla(sayi\_dizi,bas\_say\_bul(sayi));**

**fark[sayac]=buyuk\_sayi-kucuk\_sayi;**

**if(bas\_say\_bul(fark[sayac])<bas\_say\_bul(sayi))**

**fark[sayac]\*=10;**

**kontrol=dizide\_ara(fark,fark[sayac],sayac);**

**sayi=fark[sayac];**

**sayac++;**

**}while(kontrol==0);**

**printf("-----------------------------------------------------------\n");**

**if(kontrol==1)**

**{**

**printf("%d adimda seriye ulasildi.\nSerinin uzunlu: %d\nSeriyi olusturan sayilar:\n", sayac,sayac-1);**

**for(i=0;i<sayac-1;i++)**

**{**

**printf("\n%d\n",fark[i]);**

**}**

**}**

**else**

**{**

**if(kontrol==2)**

**printf("%d adimda kapali bir sayiya ulasildi. Bu kapali sayi: %d\n",sayac,fark[sayac-1]);**

**}**

**return 0;**

**}**

**void yerdegistir(int \*w,int \*t)**

**{**

**int gecici;**

**gecici=\*w;**

**\*w=\*t;**

**\*t=gecici;**

**}**

**void dizi\_sirala\_min\_max(int \*dizi,int eleman\_say)**

**{**

**int i,j,min\_indeks;**

**for(i=0;i<eleman\_say-1;i++){**

**min\_indeks=i;**

**for(j=i;j<eleman\_say;j++){**

**if(dizi[j]<dizi[min\_indeks])**

**min\_indeks=j;**

**}**

**if(min\_indeks!=i)**

**yerdegistir(&dizi[min\_indeks],&dizi[i]);**

**}**

**}**

**void dizi\_sirala\_max\_min(int \*dizi,int eleman\_say)**

**{**

**int i,j,max\_indeks;**

**for(i=0;i<eleman\_say-1;i++){**

**max\_indeks=i;**

**for(j=i;j<eleman\_say;j++){**

**if(dizi[j]>dizi[max\_indeks])**

**max\_indeks=j;**

**}**

**if(max\_indeks!=i)**

**yerdegistir(&dizi[max\_indeks],&dizi[i]);**

**}**

**}**

**int dizide\_ara(int \*dizi,int aranan,int sayac)**

**{**

**int i,kontrol=0;**

**for(i=0;i<sayac;i++){**

**if(aranan==dizi[i]){**

**if(i==sayac-1){**

**kontrol=2;**

**}**

**else{**

**kontrol=1;break;**

**}**

**}**

**}**

**return kontrol;**

**}**

**int diziden\_sayi\_hesapla(int \*dizi, int eleman\_say)**

**{**

**int i;**

**long int sonuc=0;**

**for(i=0;i<eleman\_say;i++){**

**sonuc+=dizi[i]\*us\_al(10,i);**

**}**

**return sonuc;**

**}**

**int bas\_say\_bul(long int a)**

**{**

**int basamak=0;**

**do{**

**basamak++;**

**a=a/10;**

**}while(a>0);**

**return basamak;**

**}**

**long int us\_al(int a,int b)**

**{**

**int i;**

**long int carpim=1;**

**for(i=0;i<b;i++){**

**carpim=carpim\*a;**

**}**

**return carpim;**

**}**

**5. Kullanıcı Kataloğu**

**1.** Program kullanıcıdan en az iki, en fazla dokuz basamaklı pozitif bir tam sayı istemektedir.

|  |
| --- |
|  |

**2.** Girilen sayının butun basamaklarının aynı olması durumunda program kullanıcıdan yeni bir sayı istemektedir.

|  |
| --- |
|  |

**3.** Koşullara uygun bir sayı girildiğinde gerekli hesaplamalar yapılıp sonuçlar ekrana yazdırılmaktadır.

|  |
| --- |
|  |
|  |

**5.1. Programdaki Kısıtlamalar**

**1.** Programda basamak sayısının kontrolü yapılmamıştır. Kullanıcının girdiği sayı koşullara uygun olmalıdır.

**2.** En fazla 100 adımda bir seriye ya da kapalı sayıya ulaşılacağı varsayılmıştır.