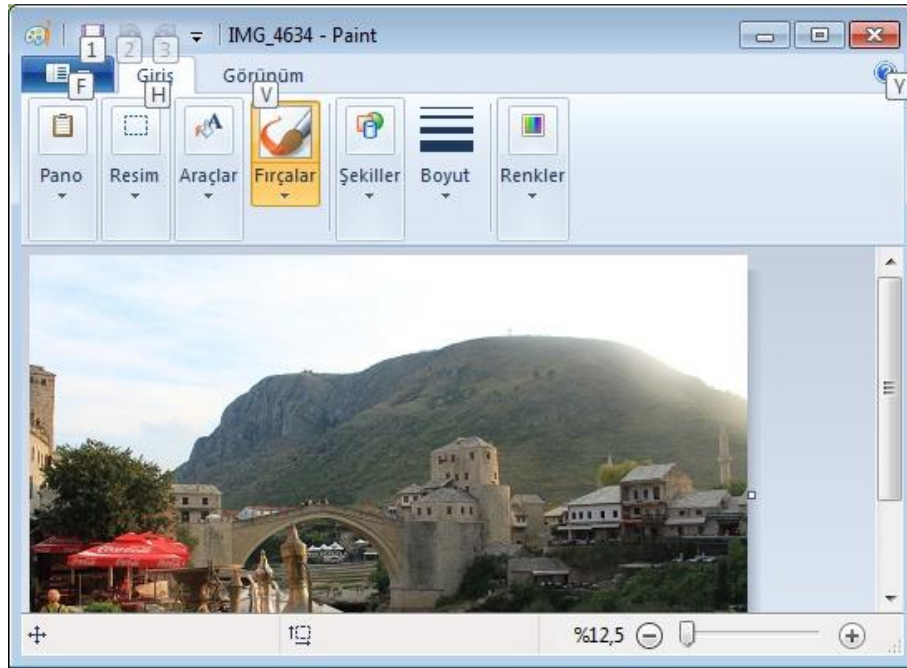


Bölüm 17. Dosya Kullanımı ve Akışlar

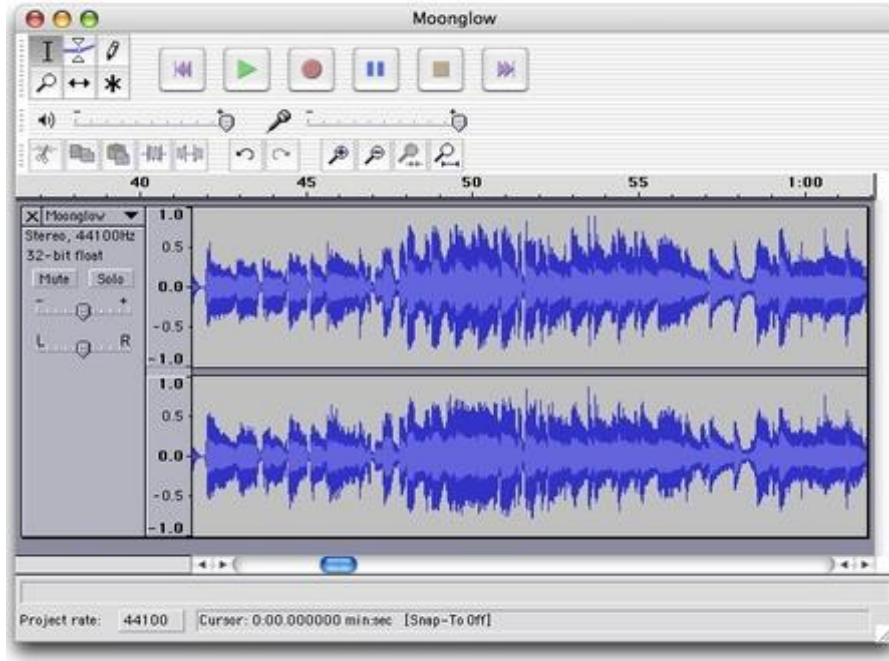
17.1 Giriş

Bilgisayarlar ile çalışan insanlar olarak verileri dosyalarda saklamanın önemini çok iyi biliyoruz. Bilgisayarın ana hafızası(RAM) bilgisayardaki sabit disklerden çok çok azdır. Yani Sabit diskler(harddisk) RAM'den daha fazla bilgi saklar. Diskler içerisindeki bilgileri enerji kesildiğinde dahi tutar. Kullanıcı istediği zaman disk üzerindeki verileri çağırarak üzerinde manipülasyonlar gerçekleştirir.

Günümüzde veritabanı yönetim sistemleri programları ile projelerimizdeki verileri daha hızlı ve rahat bir şekilde kullanabilmekteyiz. Ancak verilerimizin büyük çoğunluğu diğer dosyalarda tutulur. Resimler, grafikler, aygıt sürücüler, ses ve video verileri dosyalarda saklıdır. Dosyalar diskler üzerinde byteler dizisi olarak yer almaktadır. Bu dosyaların her birinin kendine özgü bir veri temsil düzeni bulunmaktadır. Bu düzene uyularak bu dosyalardaki veriler okunabilir, değişiklik yapılabilir ve yeniden saklanabilir. Bir dosyanın düzenini(formatını) biliyorsanız o dosyayı kolayca okuyabilir ve üzerinde değişiklik yapabilirsiniz. Dolayısıyla programlarımızda dosya işlemlerine çoğu zaman ihtiyaç duymaktayız. Şekil 17.1 de bazı dosya örnekleri verilmiştir.



a



b

Şekil 17.1 Dosya örnekleri a) resim dosyası(bmp), b) ses dosyası(wav)

Bizde program geliştirirken çoğu zaman verileri dosyalarda saklama ve onları geri okumaya ihtiyaç duyarız. c++ da dosyalarla çalışmak için bazı sınıflara ihtiyaç duyulur. Bunlar; giriş için ifstream, hem giriş hem çıkış için fstream ve çıkış için ofstream sınıflarıdır. Bu sınıfların nesneleri dosyalarla ilişkilidir ve dosyadan veri okumak ve yazmak için bu nesnelerin üye fonksiyonları kullanılır.

ifstream, istream den , fstream ise iostream den ve ofstream de ostream den türetilmiştir. Bu ata sınıflar da ios'dan türetilmiştir. ifstream, ofstream, ve fstream sınıfları FSTREAM dosyasında bildirilir.

17.1 Dosyaya formatlı bilgi yazma ve okuma

17.1.1. Dosyaya veri yazma

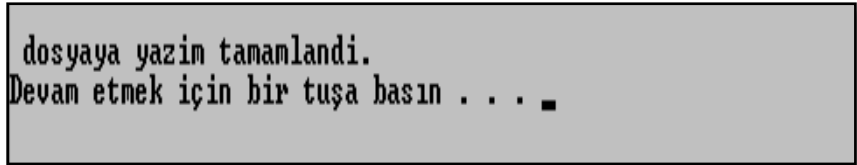
Geliştirilen programlarda, daha sonra değerlendirmek üzere bir takım verileri disk üzerinde bir dosyaya yazdırırız. Aşağıda verilen program, her hangi bir işyerinde çalışan personel ile ilgili verilerin bir kısmının kayıt edilmesini gösteren basit bir örnektir. Yani dosyaya kayıt işlemini gerçekleştiren program parçası örneğidir. Bu ve benzeri dosyalar için, dosyaya kayıt edilen verilerin çoğaltılması mümkündür. Dosyada temsil edilen verilerin çoğaltılması durumunda program yapısı değişmez. Örnek programda iki string(Personel_Adi, Personel_Soyadi), bir integer(Personel_Yas), bir double(Personel_Maas), bir karakter(Personel_cinsiyet) olmak üzere 5 değişken kullanılmıştır. Program, bu değişkenlerin değerlerini dosyaya yazar. Program çalıştırıldığında ekranda herhangi bir değişken değeri görülmeyecek, sadece dosyaya kayıt edildi mesajı alınacaktır.

Dosyaya kayıt için ofstream sınıfından DosyaYaz(yani dosyaya yaz) adında bir nesne tanımlanır. DosyaYaz nesnesi personel.txt dosyasına, yazdırma üzere tanımlanır. Dosya aktif

dizinde yeniden oluşturulur. Dosya var ise dosya yeniden oluşturulacağından dosyadaki bilgiler silinir. DosyaYaz nesnesi cout nesnesi gibidir. Yani her hangi bir değişkenin değerini, dosyaya yazmak için ekleme(<<) operatörünü kullanır. Sınıf içerisinde ekleme operatörü için operatör aşırı yükleme üye fonksiyonu vardır.

```
//  
// dosyaya kayıt-1 programı  
#include <cstdlib>  
#include <fstream>  
#include <iostream>  
#include <string>  
using namespace std;  
  
int main(int argc, char *argv[])  
{  
    string Personel_Adi= "Ali";  
    string Personel_Soyadi= "Can";  
    int Personel_Yas=34;  
    double Personel_Maas=1000.80;  
    char Personel_cinsiyet='E';  
  
    ofstream DosyaYaz("personel.txt");  
  
    DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '  
        <<Personel_Yas<<' '<<Personel_Maas<<' '  
        <<Personel_cinsiyet;  
    cout<< "dosyaya yazım tamamlandı. ";  
    system("PAUSE");  
    return EXIT_SUCCESS;  
}
```

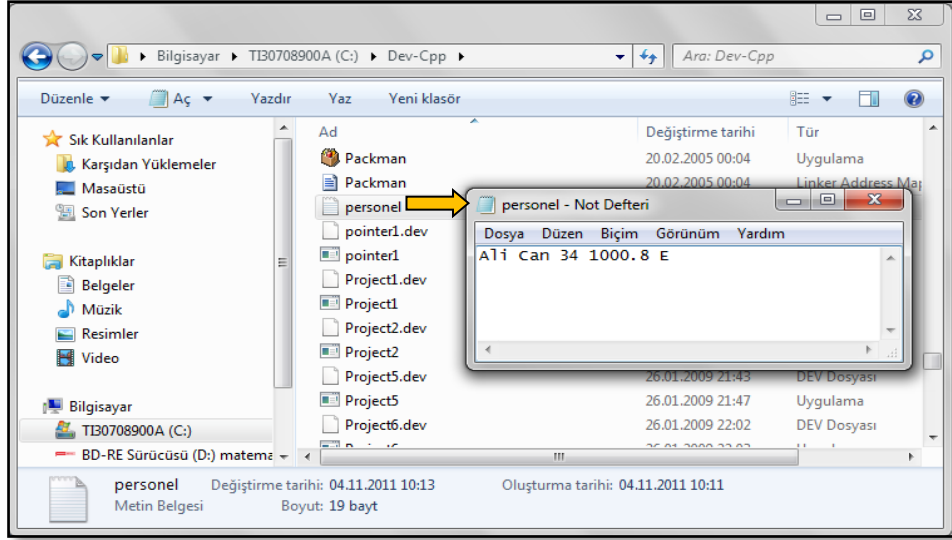
programın ekran çıktısı şekil 17.2' de verilmiştir.



```
dosyaya yazım tamamlandı.  
Devam etmek için bir tuşa basın . . . _
```

Şekil 17.2 Dosya kayıt-1 programı ekran çıktısı

Dosya aktif disk üzerindeki(harddisk), aktif dizin içerisinde yer alır. Bu dizin içerisindeki personel.txt dosyası şekilde görüldüğü üzere notpad veya benzeri text programları ile okunur ise içerisinde kayıtlı veriler görülecektir. Şekil 17.3'de Windows Gezgini ile konumu ve Not Defteri ile yazı dosyasının içeriğinin görüntülenmesi verilmiştir.



Şekil 17.3 Yazı dosyasının içeriğini "Not Defteri" ile görüntüleme

17.1.2 Dosyadan veri Okuma

ifstream nesnesini kullanarak, bir önceki programda oluşturulan dosyadan veri okuyabiliriz. Nesne oluşturulduktan sonra çıkarma-extraction(>>) operatörü kullanarak dosyadan okuma sağlanır. DosyaOku, bir ifstream sınıfı nesnesidir. Bu nesneyi ve çıkarma operatörünü kullanarak personel dosyasına daha önce yazmış olduğumuz verileri okuyalım.

```
//dosyadan okuma-1 programı
//ifstream nesnesi ve çıkarma-extraction(>>) operatörü
//kullanarak dosyadan okuma sağlanır.
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char *argv[])
{
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

    ifstream DosyaOku ("personel.txt");

    DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas>>Personel_Maas
    >>Personel_cinsiyet;

    cout<< "dosyadan okunan degerler \n";
    cout<<Personel_Adi<<endl;
    cout<<Personel_Soyadi<<endl;
    cout<<Personel_Yas<<endl;
```

```

cout<<Personel_Maas<<endl;
cout<<Personel_cinsiyet<<endl;

system("PAUSE");
return EXIT_SUCCESS;
}

```

Programın ekran çıktısı şekil 17. 4' de verilmiştir.

```

dosyadan okunan degerler
Ali
Can
34
1000.8
E
Devam etmek için bir tuşa basın . . . _

```

Şekil 17.4 dosyadan okuma-1 programı ekran çıktısı

Programda DosyaOku nesnesi, cin nesnesi ile aynı şekilde davranır. Eğer veriler dosyaya saklandığı formatta uygun olarak çağrılıyorsa(formatı uygun ise) kullanılan değişkenlere dosyadaki bu değerleri atar.

Personel dosyasına çoklu kayıt yapmak istersek, değişkenlere yeni değer atayıp, bu değişkenleri DosyaYaz nesnesi ile tekrar dosyaya yazdırmak yeterlidir. Aşağıdaki program bu kullanıma bir örnektir ve şekil 17.5 de personel dosyasının yeni hali verilmiştir.

```

// dosyaya kayıt-2 programı
// kayıt yapmak istersek, değişkenlere yeni değer atayıp,
// bu değişkenleri DosyaYaz nesnesi ile tekrar
//dosyaya yazdırmak yeterlidir
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char *argv[])
{
    string Personel_Adi= "Ali";
    string Personel_Soyadi= "Can";
    int Personel_Yas=34;
    double Personel_Maas=1000.80;
    char Personel_cinsiyet='E';

    ofstream DosyaYaz("personel.txt");

    DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
        <<Personel_Yas<<' '<<Personel_Maas<<' '
        <<Personel_cinsiyet<<"\n";

    Personel_Adi= "veli";

```

```

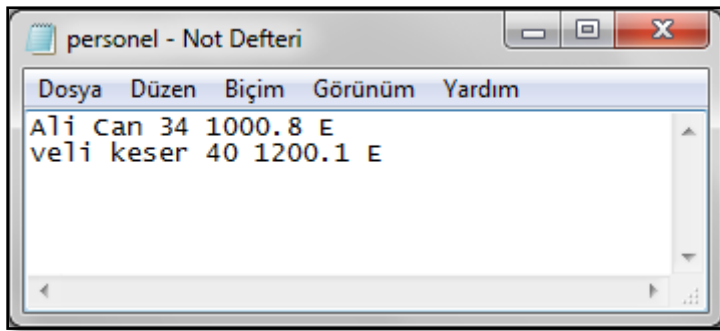
Personel_Soyadi= "keser";
Personel_Yas=40;
Personel_Maas=1200.10;
Personel_cinsiyet='E';

DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
    <<Personel_Yas<<' '<<Personel_Maas<<' '
    <<Personel_cinsiyet;

cout<< "dosyaya yazim tamamlandi. ";
system("PAUSE");
return EXIT_SUCCESS;
}

```

programı ile oluşturulan "personel" dosyasının içeriği şekil 17.5' de verilmiştir.



Şekil 17.5 dosya kayıt-2 programı ile oluşturulan "personel" dosyası içeriği

17.1.3 Dosyadan çok sayıda kayıt Okuma

Personel dosyasından çok sayıda kayıt verisi okunacak ise dosya okumak üzere açıldıktan sonra ardışıl olarak kayıtlar DosyaOku nesnesi ile okunup değişkenlere atanır. Değişken değerleride her okumada, istenir ise ekrana yazdırılarak görülebilir. Aşağıdaki program dosyadan çok sayıda kaydın okunmasına bir örnektir. Şekil 15.5' de de ekran çıktısı verilmiştir.

```

//dosyadan kayıt okuma-2 programı
// dosya okumak üzere açıldıktan sonra ardışıl olarak kayıtlar
//DosyaOku nesnesi ile okunup değişkenlere atanır.
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char *argv[])
{
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

```

```

ifstream DosyaOku ("personel.txt");

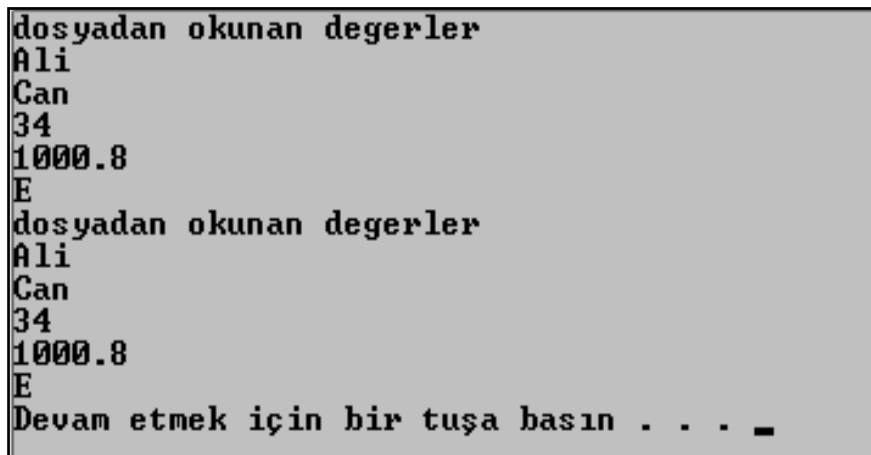
DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
>>Personel_Maas>>Personel_cinsiyet;

cout<< "dosyadan okunan degerler \n";
cout<<Personel_Adi<<endl;
cout<<Personel_Soyadi<<endl;
cout<<Personel_Yas<<endl;
cout<<Personel_Maas<<endl;
cout<<Personel_cinsiyet<<endl;
DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
>>Personel_Maas>>Personel_cinsiyet;

cout<< "dosyadan okunan degerler \n";
cout<<Personel_Adi<<endl;
cout<<Personel_Soyadi<<endl;
cout<<Personel_Yas<<endl;
cout<<Personel_Maas<<endl;
cout<<Personel_cinsiyet<<endl;

system("PAUSE");
return EXIT_SUCCESS;
}

```



```

dosyadan okunan degerler
Ali
Can
34
1000.8
E
dosyadan okunan degerler
Ali
Can
34
1000.8
E
Devam etmek için bir tuşa basın . . . _

```

Şekil 17.6 dosyadan kayıt okuma-2 programı ekran çıktısı

Gerek dosyaya birden fazla kayıt eklemekte, gerekse dosyadan çoklu kayıt okumada yukarıda verildiği şekli ile yani sıralı yapmak iyi bir yol değildir. Bir döngü ile çoklu kayıtlar ve okumalar hızlı ve kısa bir şekilde yapılabilir.

```

////dosyadan kayıt okuma-3 programı
// bir döngü ile çoklu okumalar hızlı ve kısa bir şekilde yapılabilir
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char *argv[])

```

```

{
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

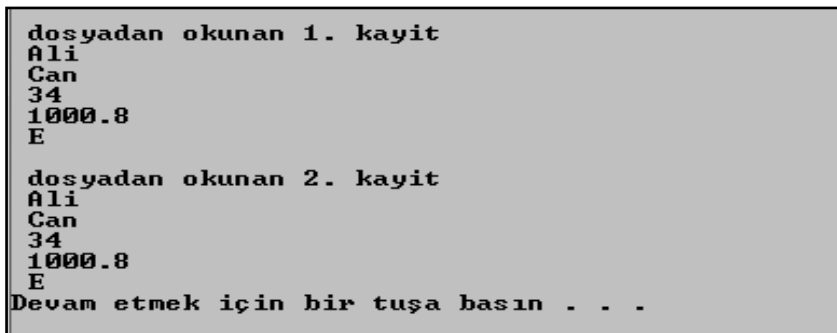
    ifstream DosyaOku ("personel.txt");
    for(int i=1;i<=2;i++)
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
        >>Personel_Maas>>Personel_cinsiyet;

        cout<< "\n dosyadan okunan "<< i<<". kayıt \n";
        cout<<Personel_Adi<<endl;
        cout<<Personel_Soyadi<<endl;
        cout<<Personel_Yas<<endl;
        cout<<Personel_Maas<<endl;
        cout<<Personel_cinsiyet<<endl;
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Programın ekran çıktısı şekil 17.7' da verilmiştir.



```

dosyadan okunan 1. kayıt
Ali
Can
34
1000.8
E

dosyadan okunan 2. kayıt
Ali
Can
34
1000.8
E
Devam etmek için bir tuşa basın . . .

```

Şekil 17.7 dosyadan kayıt okuma-3 programı ekran çıktısı

17.1.4 Dosya sonu kontrolü ile dosyadan kayıt Okuma

Çoklu okumada, dosyadaki kayıt sayısını bilemeyeceğimiz için dosya sonuna ulaşp ulaşmadığı kontrol edilmelidir. Dosya sonu kontrolü yapılmadığı durumda dosya sonuna gelindiği halde dosyadan veri okunmaya çalışılacak, bu ise hataya sebep olacaktır. **eof()** ile dosya sonu kontrolü gerçekleştirilmektedir. Aşağıdaki örnekte dosyadaki kayıtlar dosya sonu kontrolü yapılarak while döngüsü ile okunmaktadır.

```

////dosyadan kayıt okuma-4 programı
// Çoklu okumada, dosyadaki kayıt sayısını bilemeyeceğimiz için
//dosya sonuna ulaşp ulaşmadığı eof() ile kontrol edilmelidir
#include <cstdlib>
#include <fstream>

```



```

#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main(int argc, char *argv[])
{
    setlocale(LC_ALL,"turkish");
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

    int i=1;

    ifstream DosyaOku ("personel.txt");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
            >>Personel_Maas>>Personel_cinsiyet;

        cout<< "\n dosyadan okunan "<<i><<i><<". kayit \n";
        cout<<Personel_Adi<<endl;
        cout<<Personel_Soyadi<<endl;
        cout<<Personel_Yas<<endl;
        cout<<Personel_Maas<<endl;
        cout<<Personel_cinsiyet<<endl;
        i++;
    }

    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Dosyaya çoklu yazımda, kullanıcının istediği kadar kaydı girebilmesi için programda verilen değişiklikleri yapabiliriz. Burada **do ... while** döngüsü ile çıkış kontrol edilir.

```

// dosyaya kayıt-3 programı
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main(int argc, char *argv[])
{
    setlocale(LC_ALL,"turkish");
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

```

```

char cevap='e';

ofstream DosyaYaz("personel.txt");
do
{
    cout<<"\n personel adi      :"; cin>>Personel_Adi;
    cout<<"\n personel soyadi   :"; cin>>Personel_Soyadi;
    cout<<"\n personel Yasi     :"; cin>>Personel_Yas;
    cout<<"\n personel maas     :"; cin>>Personel_Maas;
    cout<<"\n personel cinsiyet :"; cin>>Personel_cinsiyet;

    DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
        <<Personel_Yas<<' '<<Personel_Maas<<' '
        <<Personel_cinsiyet<<"\n";
    cout<<"\n baska kayit yapacak misin?(e/h) ";cin>>cevap;
}while(!( cevap=='h'));

cout<< "dosyaya yazim tamamlandi. ";
system("PAUSE");
return EXIT_SUCCESS;
}

```

Programın ekran çıktısı şekil 17.8'de verilmiştir.

```

personel adi      :cemil
personel soyadi   :keser
personel Yasi     :21
personel maas     :2000.56
personel cinsiyet :e
baska kayit yapacak misin?(e/h) e
personel adi      :Ayse
personel soyadi   :cin
personel Yasi     :23
personel maas     :1800.45
personel cinsiyet :h
baska kayit yapacak misin?(e/h) h
dosyaya yazim tamamlandi. Devam etmek için bir tuşa basın . . .

```

Şekil 17.8 Dosyaya kayıt-3 programı ekran çıktısı

17.1.5 Dosya işlemlerinde menü kullanımı

Dosya işlemlerinde, kullanımı kolaylaştırmak için bir menü den faydalanılır. Bu menü vasıtası ile dosya üzerindeki farklı veri manipülasyonları ayrı ayrı ve program sonlandırmadan gerçekleştirilebilir. Bu örneğimizde dosyaya kayıt ekleme ve kayıt okuma işlemlerini ayrı ayrı gerçekleştirebilmek için menü kullanılmıştır. Verilen kod parçasında do... while döngüsü

içerisinde menü seçenekleri yazdırılmaktadır. Üç seçenek sunulmuştur. Kullanıcı bu üç seçenekten birini seçmediği müddetce döngüden çıkamayacaktır.

```
do
{
    cout << " Ne yapmak istiyorsunuz ? " << endl;
    cout << " 1) dosyaya kayıt ekleme " << endl;
    cout << " 2) dosyadan kayıt okuma " << endl;
    cout << " 3) programdan çıkış " << endl;
    cin >> secim;
} while (secim != 1 && secim != 2 && secim != 3 );
```

(1) veya (2) seçildiğinde ilgili if blokları vasıtası ile dosyaya kayıt ve dosyadan okuma gerçekleştirilecektir. Dosyaya kayıt durumunda, bir den fazla kayıt gerçekleştirilebilmesi için do ... while() döngüsü kullanılmıştır. Dosyadan okuma adeta bir listeleme şeklinde dir. Kolayca dosya içerisinde aranan bir kaydın okunması ve yazdırılması için dönüştürülebilir. Dosyadan kayıt okunduktan sonra soyadına vb alanlara göre bir if ifadesi ile kayıt karşılaştırılabilir. Koşul ifadesinin doğru sonuç vermesi durumunda okunan kayıt ekrana yazdırılır. Eğer dosya sonuna ulaşıldığı halde eşleşen kayıt bulunamamış ise de kayıt bulunamadı mesajı verilerek program sonlandırılır. Programın çalışır versiyonu aşağıda verilmiştir.

//dosyanın menü ile manipülasyonu

```
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{
    int secim,i=1;
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

    char cevap='e';

    do
    {
        cout << " Ne yapmak istiyorsunuz ? " << endl;
        cout << " 1) dosyaya kayıt ekleme " << endl;
        cout << " 2) dosyadan kayıt okuma " << endl;
        cout << " 3) programdan çıkış " << endl;
        cin >> secim;
    } while (secim != 1 && secim != 2 && secim != 3 );

    if (secim==1)
    {
        ofstream DosyaYaz("personel.txt");
        do
        {
```

```

        cout<<"\n personel adi    :"; cin>>Personel_Adi;
        cout<<"\n personel soyadi  :"; cin>>Personel_Soyadi;
        cout<<"\n personel Yasi   :"; cin>>Personel_Yas;
        cout<<"\n personel maas    :"; cin>>Personel_Maas;
        cout<<"\n personel cinsiyet :"; cin>>Personel_cinsiyet;
        DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
        cout<<"\n baska kayıt yapacak misin?(e/h) ";cin>>cevap;
        }while(!( cevap=="h"));
        cout<< "dosyaya yazım tamamlandı. ";
    }
    if(secim==2)
    {
        ifstream DosyaOku("personel.txt");
        while(!DosyaOku.eof())
        {
            DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
                >>Personel_Maas>>Personel_cinsiyet;

            cout<< "\n dosyadan okunan "<<i<<". kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;
            i++;
        }
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

programın ekran çıktısı şekil 17.9' da verilmiştir.

```

Ne yapmak istiyorsunuz ?
1) dosyaya kayıt ekleme
2) dosyadan kayıt okuma
3) programdan çıkış
2

dosyadan okunan 1. kayıt
cemil
keser
21
2000.56
e

dosyadan okunan 2. kayıt
Ayse
cin
23
1800.45
b

```

Şekil 17.9 Dosyanın menü ile manipülasyonu programı ekran çıktısı

Dosyalar üzerinde okuma ve yazma haricinde de bir takım manipülasyonlara ihtiyaç vardır. Bunlar hatalı kayıtların düzeltilmesi, yani veri güncelleme, ihtiyaç duyulmayan kayıtların silinmesi, belirli kriterlere göre dosyadan kayıt arama ve listeleme işlemleridir.

17.1.6 Dosyadan kayıt silme

Sıralı dosyalarda kayıt silmek için asıl veri dosyasının yanında birde yedek dosya kullanılır. Günlük hayatımızda basılı bir listenin yenilenmesinde, kayıtların silinmesi gerektiğinde, bu kayıtlar yeni listede yer almaz. Benzer durum dosyalarda uygulanır. Dosyadan kayıt silmek için veri dosyası okuma modunda, yedek dosyada yazma modunda açılır. Dosyadaki kayıtlardaki alanlardan(değişkenler) en az biri kullanılarak silmek istediğimiz kayıt belirtilir. Bu alan(değişken) kayıtlar için ayırt edici bir özelliğe sahip olmalıdır. Örneğin TCNO kullanılır ise çok başarılı bir ayırt edicilik sağlanmış olur. Benzer şekilde personel no, öğrenci no, üyelik no vb alanlar başarılı bir ayırt edmeye örnektir. Soyad, ad vb daha az ayırt edici alanlardır.

Silinecek kayıt için girilen değer, veri dosyasından okunan kayıt ile eşleşmiyorsa, kayıt yedek dosyaya kaydedilir. Eşleşmesi durumunda ise kayıt yedek dosyaya yazılmaz. Dosyanın sonuna gelindiğinde yedek dosya da silmek istediğimiz kayıt olmayacaktır. Bu iki dosyadan asıl veri dosyası diskten silinir(remove komutu ile) ve yedek dosyanın ismi silinmiş olan asıl dosya ismi olarak değiştirilir(rename komutu ile). Böylece istenmeyen kayıtlar silinmiş olur. Aşağıdaki kod parçası ile silme işlemi gerçekleştirilebilir.

```
if(secim==4)
{
    string ArananString;
    cout<<" silinecek kişi soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    ofstream DosyaYaz("personel.tmp");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
        >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi)
        {
            //silinecek kayıtları göster ve yedek dosyaya kayıt etme
            cout<<"\n dosyadaki kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;

        }
        else
        {
            // silinmeyecek kayıtları yedek dosyaya kaydet
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
        }
    }
    //dosyaları kapat
```

```

DosyaYaz.close();
DosyaOku.close();
// Asıl veri dosyasını diskten sil
remove( "personel.txt");
// yedek dosyanın ismini asıl dosya ismi olarak değiştir.
rename ("personel.tmp","personel.txt");
}

```

Program parçasında, yeni olarak, **close()** üye fonksiyonu, **remove**, **rename** fonksiyonlarını yer almaktadır. **close()** ile dosyalar kapatılmakta. Dosyalar kapatılmadığında **remove** ve **rename** komutları işlevlerini yerine getiremez. **remove** ile isim ve uzantısı verilen dosya disk üzerinden silinir. Dosyanın konumu verilerek istenilen disk ve klasör içerisinde dosya silinebilir. **rename** komutu ile disk üzerindeki bir dosyanın ismi değiştirilir.

17.1.7 Kayıt güncelleme(düzeltilme)

Kayıt düzeltme için de benzer bir işlem yapılır. Yani asıl veri dosyası okuma modunda, yedek dosyada yazma modunda açılır. Düzeltmek istenen kayıt okunan kayıt ile eşleştğinde kayıt için yeni veriler girilir. Yeni girilen veriler yedek dosyaya kaydedilir. Veri dosyasının sonuna ulaşıldığında, yedek dosya doğru verileri içerecektir. Silme işlemine benzer şekilde veri dosyası diskten silinir, yerine yedek dosya ismi değiştirilerek konur ise veri düzeltme işlemi tamamlanmış olur. Kayıt düzeltme için aşağıdaki kod parçası ve benzeri bir kod kullanılabilir.

```

if(secim==3)
{
    string ArananString;
    cout<<" aranacak kişi soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    ofstream DosyaYaz("personel.tmp");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
        >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi) // düzeltilecek kayıt ise
        {
            cout<<"\n dosyadaki kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;
            cout<<"\n Kayıt için yeni değerler girişi \n";
            cout<<"\n personel adi    :"; cin>>Personel_Adi;
            cout<<"\n personel soyadi   :"; cin>>Personel_Soyadi;
            cout<<"\n personel Yasi    :"; cin>>Personel_Yas;
            cout<<"\n personel maas     :"; cin>>Personel_Maas;
            cout<<"\n personel cinsiyet :"; cin>>Personel_cinsiyet;
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
        }
        else
        {
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '

```

```

        <<Personel_Yas<<' '<<Personel_Maas<<' '
        <<Personel_cinsiyet<<"\n";
    }
}
DosyaYaz.close();
DosyaOku.close();
remove( "personel.txt");
rename ( "personel.tmp","personel.txt");
}

```

17.1.8 Kayıt sorgulama

Dosya içerisinde bir kayıtlı arama işlemi silme ve kayıt düzeltmeye göre daha basit bir yapıya sahiptir. Veri dosyası okumak için açılır. Arama kriteri girilir. Yani kayıt hangi alan'a(değişkene) göre aranacaktır. Her bir okunan kayıt, arama kriteri ile karşılaştırılır. Eşleşme olduğunda kayıt ekrana yazdırılır. Eşleşen kayıt aranan kayıt ise arama sonlandırılır, değil ise aramaya devam edilebilir. Dosya sonuna ulaşıldığı halde eşleşen bir kayıt yoksa aranan kayıt dosyada yoktur. Arama ile ilgili örnek bir kod aşağıda verilmiştir.

```

if(secim==5)
{
    string ArananString;
    cout<<" aranacak kişi soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
        >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi)
        {
            cout<<"\n Bulunan kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;
            system("PAUSE");
            exit(1);
        }
    }
}

```

17.1.9 Basitleştirilmiş sıralı dosya örneği

Programda, dosya üzerinde gerçekleştirebilecek kayıt, arama, silme, düzeltme manipülasyonları verilmiştir.

```

//sıralı dosya örneği
#include <windows.h>
#include <fstream>
#include <iostream>
#include <stdio.h>

```

```

#include <string>

using namespace std;

int main(int argc, char *argv[])
{
    int secim,i=1;
    string Personel_Adi;
    string Personel_Soyadi;
    int Personel_Yas;
    double Personel_Maas;
    char Personel_cinsiyet;

    char cevap='e';

    do
    {
        cout << " Ne yapmak istiyorsunuz ? " << endl;
        cout << " 1) dosyaya kayıt ekleme " << endl;
        cout << " 2) dosyadan kayıt okuma " << endl;
        cout << " 3) dosyadan kayıt düzeltme " << endl;
        cout << " 4) dosyadan kayıt silme " << endl;
        cout << " 5) kayıt arama " << endl;
        cout << " 6) programdan çıkış " << endl;
        cin >> secim;
    } while (secim != 1 && secim != 2 && secim != 3 && secim != 4 && secim != 5 && secim != 6);

    if (secim==1)
    {
        ofstream DosyaYaz("personel.txt");
        do
        {
            cout<<"\n personel adi    :"; cin>>Personel_Adi;
            cout<<"\n personel soyadi  :"; cin>>Personel_Soyadi;
            cout<<"\n personel Yasi    :"; cin>>Personel_Yas;
            cout<<"\n personel maas    :"; cin>>Personel_Maas;
            cout<<"\n personel cinsiyet :"; cin>>Personel_cinsiyet;
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
            cout<<"\n baska kayıt yapacak misin?(e/h) ";cin>>cevap;
        }while(!( cevap=='h'));
        cout<< "dosyaya yazım tamamlandı. ";
    }
    if(secim==2)
    {
        ifstream DosyaOku("personel.txt");
        while(!DosyaOku.eof())
        {
            DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
                >>Personel_Maas>>Personel_cinsiyet;

            cout<< "\n dosyadan okunan " <<i<<" . kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;

```



```

        cout<<Personel_Yas<<endl;
        cout<<Personel_Maas<<endl;
        cout<<Personel_cinsiyet<<endl;
        i++;
    }
}
if(secim==3)
{
    string ArananString;
    cout<<" aranacak kiři soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    ofstream DosyaYaz("personel.tmp");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
            >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi)
        {
            cout<<"\n dosyadaki kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;
            cout<<"\n Kayıt için yeni değerkler giriři \n";
            cout<<"\n personel adi    :"; cin>>Personel_Adi;
            cout<<"\n personel soyadi  :"; cin>>Personel_Soyadi;
            cout<<"\n personel Yasi    :"; cin>>Personel_Yas;
            cout<<"\n personel maas    :"; cin>>Personel_Maas;
            cout<<"\n personel cinsiyet :"; cin>>Personel_cinsiyet;
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
        }
        else
        {
            DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
                <<Personel_Yas<<' '<<Personel_Maas<<' '
                <<Personel_cinsiyet<<"\n";
        }
    }
    DosyaYaz.close();
    DosyaOku.close();
    remove("personel.txt");
    rename("personel.tmp","personel.txt");
}
if(secim==4)
{
    string ArananString;
    cout<<" silinecek kiři soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    ofstream DosyaYaz("personel.tmp");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
            >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi)

```

```

    {
        cout<< "\n dosyadaki kayıt \n";
        cout<<Personel_Adi<<endl;
        cout<<Personel_Soyadi<<endl;
        cout<<Personel_Yas<<endl;
        cout<<Personel_Maas<<endl;
        cout<<Personel_cinsiyet<<endl;

    }
    else
    {
        DosyaYaz <<Personel_Adi<<' '<<Personel_Soyadi<<' '
            <<Personel_Yas<<' '<<Personel_Maas<<' '
            <<Personel_cinsiyet<<"\n";
    }
}
DosyaYaz.close();
DosyaOku.close();
remove( "personel.txt");
rename ("personel.tmp","personel.txt");
}
if(secim==5)
{
    string ArananString;
    cout<<" aranacak kişi soyadı :";cin>>ArananString;
    ifstream DosyaOku("personel.txt");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Personel_Adi>>Personel_Soyadi>>Personel_Yas
            >>Personel_Maas>>Personel_cinsiyet;
        if(ArananString==Personel_Soyadi)
        {
            cout<< "\n Bulunan kayıt \n";
            cout<<Personel_Adi<<endl;
            cout<<Personel_Soyadi<<endl;
            cout<<Personel_Yas<<endl;
            cout<<Personel_Maas<<endl;
            cout<<Personel_cinsiyet<<endl;
            system("PAUSE");
            exit(1);
        }
    }

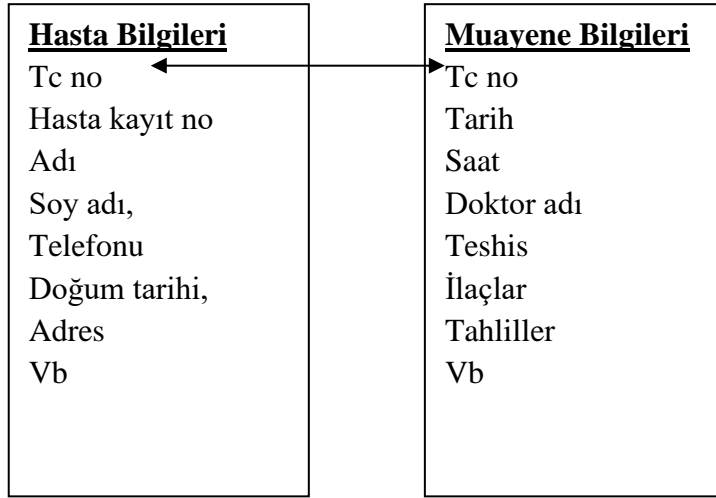
}
system("PAUSE");
return EXIT_SUCCESS;
}

```

17.2 Çoklu dosya yönetimi

Uygulamada tek bir dosya kullanımı genellikle yetersiz kalmaktadır. Gerek veri tekrarını önlemek, gereksede veri organizasyonunu sağlamak üzere çok sayıda ve bir biri ile ilişkili dosyalarda verilerimizi tutarız. Örneğin hastaların kayıtlarının tutulmasında hastanın değişmeyen ad, soy ad, tel, doğum tarihi vb. bilgileri yanında, hastanın her muayene olduğundaki bilgileri vardır. Muayene bilgileri hasta için sık değişen bir veridir. Ayı bir

dosyada tutulması zorluluk gerektirir. Bu durumda en azından iki farklı dosya ile veri manipülasyonu gerçekleştirilmelidir. Şekil de dosyalar için muhtemel alanlar verilmiştir.



(TC NO) her iki dosyadada yer almaktadır. Bu bir veri tekrarı gibi görünsede bize iki dosya arasındaki ilişkiyi kurmamızı sağlamada yardımcı olacaktır. Örneğin, Hastanın genel bilgilerini bir kere kayıt ettiğimizde, hasta her seferinde muayeneye geldiğinde sadece muayene bilgileri dosyasına muayene ile ilgili bilgiler kayıt eklenecektir. Kime ait olduğuda TC NO ile bilinecektir. Yani iki dosya arasında bilgilere ulaşımı TC no bilgisi sağlayacaktır.

```
//çok sayıda ilişkili dosya kullanımı
#include <windows.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{ // hasta.txt dosyasında yer alacak alanlar
    int secim,i=1;
    string Hasta_Adi;
    string Hasta_Soyadi;
    string Tc;
    string Tel;
    int Hasta_Yas;
    char Hasta_cinsiyet;
    //muayene.txt dosyasında yer alacak alanlar
    //string Tc; iki dosya arasındaki ilişkiyi sağlamak için
    string tarih;
    string DK_Adi;
    string Teshis;
    double Ucret;

    char cevap='e';

    do
```

```

{
    cout << " Ne yapmak istiyorsunuz ? " << endl;
    cout << " 1) Hasta kayıt ekleme " << endl;
    cout << " 2) Randevu kayıt " << endl;
    cout << " 3) programdan çıkış " << endl;
    cin >> secim;
} while (secim != 1 && secim != 2 && secim != 3 );

if (secim==1)
{
    ofstream DosyaYaz;
    DosyaYaz.open("Hasta.txt",ios::app);
    do
    {
        cout<<"\n Hasta TC      :"; cin>>Tc;
        cout<<"\n Hasta adi      :"; cin>>Hasta_Adi;
        cout<<"\n Hasta soyadi    :"; cin>>Hasta_Soyadi;
        cout<<"\n Hasta Yasi      :"; cin>>Hasta_Yas;
        cout<<"\n telefonu      :"; cin>>Tel;
        cout<<"\n cinsiyet      :"; cin>>Hasta_cinsiyet;

        DosyaYaz <<Tc<<" "<<Hasta_Adi<<' '<<Hasta_Soyadi<<' '
            <<Hasta_Yas<<' '<<Tel<<' '
            <<Hasta_cinsiyet<<"\n";
        cout<<"\n baska kayit yapacak misin?(e/h) ";cin>>cevap;
    }while(!( cevap=='h'));
    DosyaYaz.close();
    cout<< "dosyaya yazim tamamlandi. ";
}

if(secim==2)
{
    string TC;
    cout<<" Hasta TC :";cin>>TC;
    ifstream DosyaOku("Hasta.txt");
    while(!DosyaOku.eof())
    {
        DosyaOku >>Tc>>Hasta_Adi>>Hasta_Soyadi
            >>Hasta_Yas>>Tel>>Hasta_cinsiyet;
        if(TC==Tc)
        {
            cout<< "\n Hasta Bilgileri \n";
            cout<<" TC no      :"<<Tc<<endl;
            cout<<"adi        :"<<Hasta_Adi<<endl;
            cout<<"soy adi     :"<<Hasta_Soyadi<<endl;
            cout<<"Yaşı       :"<<Hasta_Yas<<endl;
            cout<<"telefonu    :"<<Tel<<endl;
            cout<<"cinsiyeti:"<<Hasta_cinsiyet<<endl;
            cout<<" randevu istiyor mu? (e/h)"<<endl;
            cin>>cevap;
            if (cevap=='e')
            {
                DosyaOku.close();
                ofstream DosyaYaz;
                DosyaYaz.open("Muayene.txt",ios::app);
                cout<<"\n Randevu Bilgileri \n";
                cout<<"\n Tarih          :"; cin>>tarih;
            }
        }
    }
}

```



```

cout<<setw(11)<<" TC " <<setw(10)<<" Tarih " <<setw(12)<<" Doktor Adi " <<setw(10)
<<" Teshis " <<setw(6)<<" Ucret" <<endl;
while(!DosyaOku.eof())
{

DosyaOku>>Tc>>tarih>>DK_Adi>>Teshis>>Ucret;

if(TC==Tc)
{
cout<<setw(11)<<Tc<<setw(10)<<tarih<<setw(12)<<          DK_Adi<<setw(10)          <<
Teshis<<setw(6)<< Ucret<<endl;
}
}
DosyaYaz.close();
system("PAUSE");
exit(1);
}
}
}

```

program kullanımını yansıtır örnek ekran çıktısı şekil 17.10’da verilmiştir.

```

Ne yapmak istiyorsunuz ?
1> Hasta kayıtlı ekleme
2> Randevu kayıtlı
3> Muayene sorgula
3
Hasta TC :14
Hasta Bilgileri
TC no      :14
adi        :cem
soy adi    :oz
Yaşı       :32
telefonu   :264
cinsiyeti  :e
randevu bilgileri
          TC      Tarih  Doktor Adi    Teshis  Ucret
          14      17      murat      grip    20.5
          14      19      fatma      ishal    20.5
          1406/08/2013  Aysel      grip     30
          1406/08/2013  Aysel      grip     30
Devam etmek için bir tuşa basın . . .

```

Şekil 17.10 Hasta muayene sorgulama örnek ekran çıktısı

17.3. Boşluk içeren string veriler

Boşluk içeren karakter dizinlerini okuyabilmek için, kaydederken yeni satır manipülatörü “\n” kullanılarak satır atlatılmalı, okunurken ofstream nesnesi ile direk okunamaz, getline üye fonksiyonu kullanılarak satır satır okuma yapılabilir.

```

//Dosyaya Kayıt-4 programı
//boşluk içeren karakter dizinlerinin dosyaya kayıt edilmesi
#include <fstream>
#include <iostream>
#include <string>

```

```

using namespace std;

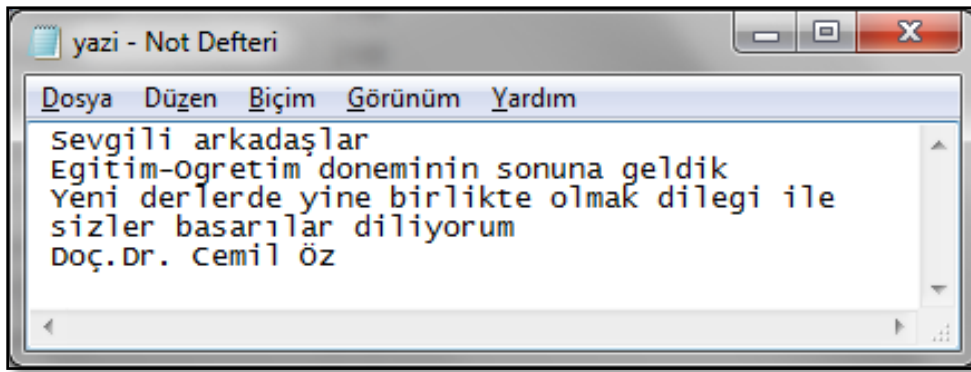
int main(int argc, char *argv[])
{
    ofstream DosyaYaz("yazi.txt");

    DosyaYaz<<" Sevgili arkadaşlar  \n";
    DosyaYaz<<" Egitim-Ogretim doneminin sonuna geldik \n";
    DosyaYaz<<" Yeni derlerde yine birlikte olmak dilegi ile \n";
    DosyaYaz<<" sizlere basarılar diliyorum \n";
    DosyaYaz<<" Doç.Dr. Cemil Öz \n";

    cout<< "dosyaya yazım tamamlandı. ";
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

program çalıştırıldığında şekil 17.11' da içeriği verilen yazi dosyası oluşturulacaktır.



Şekil 17.11 dosya kayıt-4 programı ile oluşan "yazi" dosyası içeriği

17.4 getline() fonksiyonu ile dosyadan satır satır okuma

Karakter dizini dosyadan okumak için ifstream oluşturulur ve ifstream' in üye, getline() fonksiyonu ile her seferinde bir satır okunur. Fonksiyon her seferinde boşluklarda dahil olmak üzere karakterleri satır sonuna kadar ("\\n" sezene kadar) okur. Okunan karakter dizini, ikinci parametre olarak belirtilen eleman sayısını aşmamak kaydı ile karakter dizinine aktarılır.

```

// dosyayadan kayıt okuma-4
//boşluklar içeren karakter dizini okuma
//getline() fonksiyonu ile gerçekleştirilebilir.
//klavyeden atamaya benzer şekilde iki parametre alır.
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char *argv[])
{

```

```

char metin[80];

ifstream DosyaOku("yazi.txt");

while(!DosyaOku.eof())
{
    DosyaOku.getline(metin, 80);
    cout<<metin<<endl;
}

cout<< "dosyadan okuma tamamlandi. ";
system("PAUSE");
return EXIT_SUCCESS;
}

```

Dosya kayıt-4 programından sonra, programımız çalıştırıldığında şekil 17.12' da verilen ekran çıktısı elde edilir.

```

Sevgili arkadaşlar
Eğitim-Öğretim döneminin sonuna geldik
Yeni derlerde yine birlikte olmak dileği ile
sizler başarılar diliyorum
Doç.Dr. Cemil Öz

dosyadan okuma tamamlandi. Devam etmek için bir tuşa basın . . .

```

Şekil 17.12 dosyada kayıt okuma-4 programı ekran çıktısı.

Dosyaya nesnelerin kayıt edilmesinde, biraz daha dikkatli olunması gerekmektedir. Aşağıdaki örnekte sınıf üye değişkenleri public olarak tanımlanmıştır. Dolayısı ile sınıfın dışında bu üye değişkenlere erişilebilmiştir. Private verilmesi durumunda üye değişkenlere erişilemeyeceğinden program çalışmayacaktır.

```

//sınıflarda dosya örneği(personel) programı
#include <cstdlib>
#include <iostream>
#include <string>
#include <locale.h>
#include <fstream>
#include <iostream>

using namespace std;
const int boyut=20;

class Personel
{
public:
    string Personel_Adi;
    string Personel_Soyadi;
    float Personel_Maas;
    int Personel_Yas;

    void deger_ata()

```



```

    {
        cout<<"\n Adı  :";
        cin>>Personel_Adi;
        cout<<"\n Soyadı :";
        cin>> Personel_Soyadi;
        cout<<"\n maaş  :";
        cin>> Personel_Maas;
        cout<<"\n yaşı  :";
        cin>> Personel_Yas;
    }

    // sınıfın dışında yazılacak fonksiyon bildirimleri
    void yazdir();
};
void Personel::yazdir()
{
    cout<<"\n Adı  : "<<Personel_Adi<<endl;
    cout<<"\n Soyadı : "<<Personel_Soyadi<<endl;
    cout<<"\n maaş  : "<<Personel_Maas<<endl;
    cout<<"\n yaşı  : "<<Personel_Yas<<endl;
}

void KlavyedenOkuDosyayaYaz(int ps)
{
    ofstream dosya("personel.txt");
    for (int i = 1; i<=ps; i++)
    {
        Personel p;
        p.deger_ata();
        dosya << p.Personel_Adi << " " << p.Personel_Soyadi << " "
        << p.Personel_Maas << " " <<p.Personel_Yas<< endl;
    }
    dosya.close();
}

void DosyadanOkuekranaYaz(int ps)
{
    ifstream dosya("personel.txt");
    for (int i = 1; i<=ps; i++)
    {
        Personel p;
        dosya >>p.Personel_Adi>>p.Personel_Soyadi
        >>p.Personel_Maas>>p.Personel_Yas;
        p.yazdir();
    }
    dosya.close();
}

int main(int argc, char *argv[])
{
    setlocale(LC_ALL,"turkish");
    int secim;
    do
    {
        cout << "  Ne yapmak istiyorsunuz ? " << endl;
        cout << "  1) kayıt ekleme " << endl;
    }

```

```

        cout << " 2) kayıt oku " << endl;
        cout << " 3) programdan çıkış " << endl;
        cin >> secim;
    } while (secim != 1 && secim != 2 && secim != 3 );
    if(secim==1)
    {
        int ps;
        cout << "Personel Sayısı:"; cin >> ps;
        KlavyedenOkuDosyayaYaz(ps);
    }
    if(secim==2)
    {
        int ps;
        cout << "Personel Sayısı:"; cin >> ps;
        DosyadanOkuekranaYaz(ps);
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

17.5 İkili ve rasgele erişimli dosyalar

Buraya kadar verilen dosya örneklerinde text türünden dosyalar kullandık. Bu dosyaların her hangi bir text editörü ile okuna bileceğini belirtmiştik. Bu bölümde İkili(binary) dosyalar üzerinde duracağız. İkili dosyalarda ekleme(>>) ve çıkarma(<<) operatörleri ve getline() benzeri fonksiyonlar verimli değillerdir. Veriler arasındaki boşluk vb düzenlemelere ihtiyaç duyulmaz. Binary dosyalarda veriler hafızada saklandığı şekli ile sıralı bir düzende dosyaya kaydedilir. Text editörleri ile içeriğini görmek mümkün değildir. Binary dosyalar okuma yapmak için ifstream ve yazma yapmak için ofstream ile kullanılabilir. Dosya hem okuma, hemde yazma için açılacak ise fstream sınıfı nesneleri ile kullanılabilir. Dosyaya kayıt için write() ve dosyadan okumak için read() üye fonksiyonları kullanılır. Bu fonksiyonlarda veriler char tipindedir. Yani byte dizisi halindedir. İki parametreye sahiptir.

```

write(hafıza bloğu, boyut);
read(hafızabloğu, boyut);

```

Burada hafıza bloğu char dizisine işaret eden bir işaretçidir(char*) ve byte dizisinin adresine işaret eder. Boyut ise tamsayı türünden bir değişkendir. Okunacak veya yazılacak karakterin sayısını belirtir. Nesne, structure vb veriler reinterpret_cast operatörü ile write ve read fonksiyonları ile kullanılacak formata dönüştürülebilir.

Reinterpret_cast operatörü

Veri türleri arasında dönüşüm için kullanılan birkaç ifadeden biride reinterpret_cast fonksiyonudur. Aşağıdaki programda integer elemanlardan oluşan a dizisi, dosyaya binary formatında kaydedilmek ve okunmak üzere dönüştürülmektedir. Programda;

```

DosyaYaz.write(reinterpret_cast<char*>(a),10*sizeof(int));

```

İfadesi ile dizi char tipine dönüştürülmekte ve daha sonrada dosyaya kayıt edilmektedir. Dizide 10 eleman bulunmakta her bir eleman 4 byte hafıza kaplamaktadır. **10*sizeof(int)** ifadesi ile dizinin boyutu hesaplanmaktadır (burada 40 byte).

```
//reinterprint_cast
#include <windows.h>
#include <fstream>
#include <iostream>

#include <iomanip>

using namespace std;

int main(int argc, char *argv[])
{ // hasta.txt dosyasında yer alacak alanlar
  int a[10]={ 10,12,8,20,30,10,9,7,1,10};
  ofstream DosyaYaz;
  DosyaYaz.open("dizi.dat",ios::binary);
  DosyaYaz.write(reinterpret_cast<char*>(a),10*sizeof(int));
  DosyaYaz.close();
  // dizi eleman değerlerini sıfırla
  // böylece dosyadan okumanın doğruluğu kontrol edilir.
  for(int i=0; i<10; i++)
    a[i]=0;
  ifstream DosyaOku("dizi.dat",ios::binary);
  DosyaOku.read(reinterpret_cast<char*>(a),10*sizeof(int));
  for(int i=0; i<10; i++)
    cout<<a[i]<<" ";

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Benzer şekilde nesnelerinde diske yazılması ve diskten okunması için **reinterpret_cast** ifadesini kullanabiliriz. Bu bize birçok kolaylık sağlar. Aşağıdaki programda, personel sınıfı nesnesinin dosyaya ikili düzende kayıt edilmesi ve okunması örneği gösterilmiştir.

```
//reinterprint_cast ile nesne kullanımı
//sınıflarda dosya örneği(personel) programı
#include <cstdlib>
#include <iostream>
#include <string>
#include <locale.h>
#include <fstream>
#include <iostream>

using namespace std;

class Personel
{
private:
  char Personel_Adi[10];
  char Personel_Soyadi[15];
  float Personel_Maas;
```

```

        int Personel_Yas;
public:

void deger_ata()
{
    cout<<"\n Adı  :";
    cin>>Personel_Adi;
    cout<<"\n Soyadı :";
    cin>> Personel_Soyadi;
    cout<<"\n maaş  :";
    cin>> Personel_Maas;
    cout<<"\n yaşı  :";
    cin>> Personel_Yas;
}

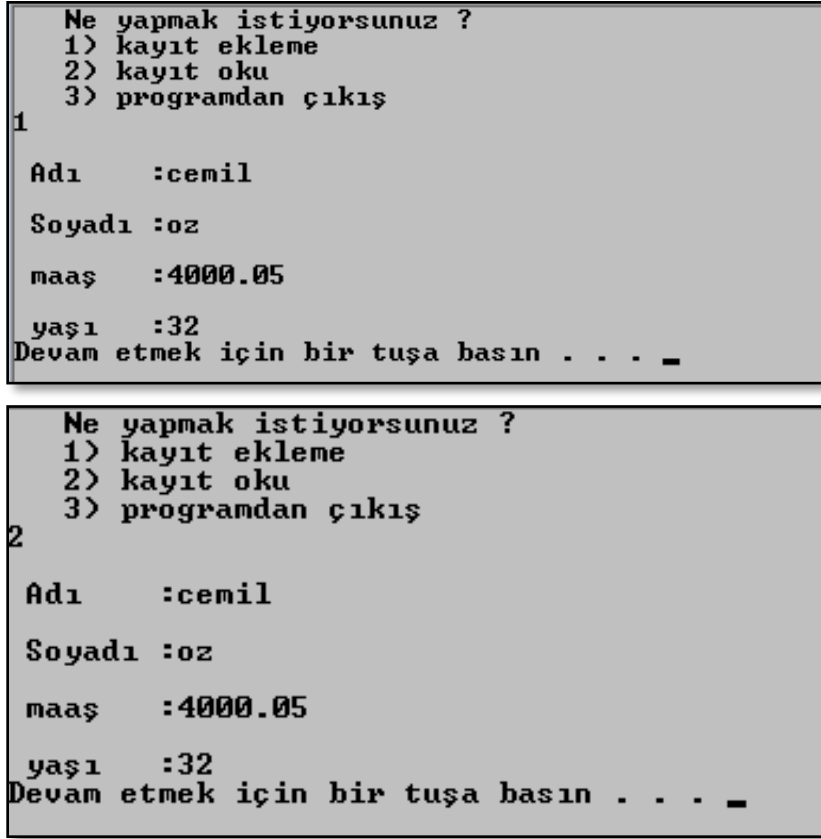
// sınıfın dışında yazılacak fonksiyon bildirimleri
void yazdir();
};
void Personel::yazdir()
{
    cout<<"\n Adı  :"<<Personel_Adi<<endl;
    cout<<"\n Soyadı :"<<Personel_Soyadi<<endl;
    cout<<"\n maaş  :"<<Personel_Maas<<endl;
    cout<<"\n yaşı  :"<<Personel_Yas<<endl;
}

int main(int argc, char *argv[])
{
    setlocale(LC_ALL,"turkish");
    Personel P1;
    int secim;
    do
    {
        cout << "  Ne yapmak istiyorsunuz ? " << endl;
        cout << "  1) kayıt ekleme " << endl;
        cout << "  2) kayıt oku  " << endl;
        cout << "  3) programdan çıkış " << endl;
        cin >> secim;
    } while (secim != 1 && secim != 2 && secim != 3 );
    if(secim==1)
    {
        ofstream DosyaYaz("personel.dat", ios::out|ios::binary);
        P1.deger_ata();
        DosyaYaz.write(reinterpret_cast<char*>(&P1),sizeof(Personel));
        DosyaYaz.close();
    }
    if(secim==2)
    {
        ifstream DosyaOku("personel.dat", ios::in|ios::binary);
        DosyaOku.seekg(0);
        DosyaOku.read(reinterpret_cast<char*>(&P1),sizeof(Personel));
        P1.yazdir();
        DosyaOku.close();
    }
}

```

```
system("PAUSE");  
return EXIT_SUCCESS;  
}
```

Programın ekran çıktısı şekil 17.13 de verilmiştir.



```
Ne yapmak istiyorsunuz ?  
1> kayıt ekleme  
2> kayıt oku  
3> programdan çıkış  
1  
Adı      :cemil  
Soyadı   :oz  
maaş     :4000.05  
yaş1     :32  
Devam etmek için bir tuşa basın . . . _  
  
Ne yapmak istiyorsunuz ?  
1> kayıt ekleme  
2> kayıt oku  
3> programdan çıkış  
2  
Adı      :cemil  
Soyadı   :oz  
maaş     :4000.05  
yaş1     :32  
Devam etmek için bir tuşa basın . . . _
```

Şekil 17.13 ikili dosyalara Nesne kaydı ve okunması

17.5.1 Open() fonksiyonu

Open fonksiyonu dosyaları açmak için kullanılır. fsream, ofstream veya ifstream sınıflarından bir dosya nesnesi tanımlandığında open fonksiyonu kullanılarak dosya açılabilir. Open fonksiyonu parametre olarak dosya konumu ve ismi ile dosyanın ne amaçla açıldığını belirtir dosya modu parametrelerini alır.

Mod parametresinin üyeleri olarak; in, out, ate, ape, binary vb ni sayabiliriz. Bu üyelerin işlevleri sırası ile;

- in** : dosyayı okuma modunda açmak,
- out** : dosyayı yazma modunda açmak,
- app** : dosyayı ilave kayıt yapmak için açmak,(ilave kayıt dosyanın sonuna yapılacaktır.)
- ate** : dosya sonundan okumaya ve yazmaya başla,
- binary** : dosyayı ikili moda açma,

Olarak verilebilir.

17.5.2 Rasgele erişimli dosyalar

Sıralı erişimli dosyalar, çoğu uygulama için uygun değildir. Kayıtlara erişim sıra iledir. Okumaya birinci kayıttan başlanıp sıra ile istenilen kayıt olup olmadığı test edilir. Bu ise zaman kaybına sebep olur. Rasgele erişimde ise sıra ile değil doğrudan bir erişim söz konusudur. Banka uygulamaları başta olmak üzere birçok uygulamada rasgele erişim uygulanır. Biz rasgele erişim ve sıra ile erişimi, orta öğrenimimizdeki sözlülerden çok iyi biliriz. Öğretmen sözlüyü sınıf listesine göre yapıyorsa genellikle listenin sonundakiler muhtemelen o derste sıranın kendilerine gelmeyeceğini veya sıra kendilerine gelene kadar, bir hatırlama süresine sahip olacaklarından biraz daha rahat olacaklardır. Rasgele olduğunda, öğretmen her hangi bir öğrenciyi kaldırabilir. Bu yüzden her öğrenci tedirgin olacaktır.

Örnek programda rasgele erişimli ve ikili kayıt yapısında dosya uygulaması verilmiştir. Programda Personel sınıfı kullanılmıştır. Personel sınıfının üye fonksiyonları ve üye değişkenleri aşağıda verilmiştir.

```
class Personel
{
private:
    int personelNo;
    char soyadi[ 15 ];
    char adi[ 10 ];
    double maas;

public:
// kurucu fonksiyon
Personel( int = 0, const string & = "", const string & = "", double = 0.0 );

// Personelno için fonksiyonlar
void setpersonelno( int );
int getpersonelno () const;

// Personel adı için fonksiyonlar
void setadi( const string & );
string getadi () const;

// Personel soyadı için fonksiyonlar
void setsoyadi( const string & );
string getsoyadi () const;

// Personel Maas için fonksiyonlar
void setmaas( double maasD );
double getmaas() const;

}; //sınıfın sonu

Personel::Personel( int PersonelnoD, const string &adiD, const string &soyadiD, double maasD ) :
personelNo(PersonelnoD),maas(maasD)
{
    setsoyadi(soyadiD );
    setadi( adiD );
} // kurucu fonksiyon sonu
```

```
// personelno gir
int Personel::getpersonelno() const
{
    return personelNo;
} // getmaas fonksiyonunun sonu

// Maas deęeri
void Personel::setpersonelno( int personelnoD )
{
    personelNo =personelnoD;
} // setmaas fonksiyonunun sonu

// getsoyadi fonksiyonu
string Personel::getsoyadi() const
{
    return soyadi;
} // getsoyadi fonksiyonunun sonu

// setsoyadi fonksiyonu
void Personel::setsoyadi( const string &soyadiD )
{
    // soyadının en fazla 15 karakter olmasını sağla
    int ksay = soyadiD.size();
    ksay = ( ksay < 15 ? ksay : 14 );
    soyadiD.copy( soyadi, ksay );

    soyadi[ ksay ] = '\0'; // soyadının sonuna null karakteri ekle
} // setsoyadi fonksiyonu sonu

// getadi fonksiyonu
string Personel::getadi() const
{
    return adi;
} // getadi fonksiyonu sonu

// setadi fonksiyonu
void Personel::setadi( const string &adiD )
{
    // adın en fazla 10 karakter olmasını sağla
    int ksay = adiD.size();
    adiD.copy(adi,ksay);
    ksay = ( ksay < 10 ? ksay : 9 );
    adi[ksay ] = '\0'; // null karakteri ekle
} // setadi fonksiyonu sonu

// getmaas fonksiyonu
double Personel::getmaas() const
{
    return maas;
} // getmaas fonksiyonu

// setmaas fonksiyonu
void Personel::setmaas( double maasD )
{
    maas =maasD;
```

```
}// setmaas fonksiyonu sonu
```

Dosya üzerinde yapılabilecek manipülasyonlar çok çeşitli olduğundan dolayı programın bir menu ile kullanılması gerekir. Böylece dosya üzerindeki farklı manipilasyonlar, seçime bağlı olarak yapılabilecektir. Program için geliştirilen menu aşağıda verilmiştir.

```
int secim;
do
{
    cout << " Ne yapmak istiyorsunuz ? " << endl;
    cout << " 1) dosya oluşturun " << endl;
    cout << " 2) kayıt ekle " << endl;
    cout << " 3) kayıtları listele " << endl;
    cout << " 4) Kayıt ara " << endl;
    cout << " 5) Kayıt düzelt " << endl;
    cout << " 6) Kayıt sil " << endl;
    cin >> secim;
} while (secim != 1 && secim != 2 && secim != 3 && secim != 4&& secim != 5&& secim != 6);
```

Rasgele erişimli dosyadaki kayıt sayısının bilinmesi ve kayıt uzunluğunun sabit olması gerekir. Böylece dosyadaki ulaşılacak istenen kayıt'ın konumu kolayca hesaplanabilir. Verilen programda dosya oluştururken kolaylık olması için dosyadaki kayıt sayısı belirlenmiş ve her bir kayıt için boş kayıt girilmiştir. Programda 100 adet kayıt ön görülmüş ve aşağıdaki kod parçası ile boş kayıtlar girilmiştir.

```
if(secim==1)
{
    ofstream DosyaYaz( "personel.dat", ios::out | ios::binary );
    // dosya açılmıyor ise çık
    if (!DosyaYaz )
    {
        cerr << "dosya açılmıyor" << endl;
        exit(0 );
    } // end if

    Personel Per;

    // dosyaya 100 adet boş kayıt ekle
    for ( int i = 0; i < 100; ++i )
        DosyaYaz.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
}
```

17.5.2. 1 Rasgele erişimli dosyaya kayıt ekleme

Dosyaya kayıt eklemek için eklenecek personel no bizim kayıtlarımızın yerini belirlemekte, bir personel nesnesinin boyutu sabit ve bilindiğine göre(sizeof(Personel)), kayıtların yapılacağı dosya pozisyonu((Per.getpersonelno() - 1) *sizeof(Personel)) belirlenir ve dosya kayıt işaretçisi belirlenen konuma işaret eder. Bu işlevi gerçekleştiren kod parçası ise

```
if(secim==2)
{
    fstream DosyaYaz( "personel.dat", ios::in | ios::out | ios::binary );
```



```

// eğer dosya açılmıyor ise çık
if (!DosyaYaz )
{
cerr << "Dosya açılmıyor." << endl;
exit( EXIT_FAILURE );
} // end if
cout << " Personel No Gir (1 to 100)\n? ";

// Personel no gir
Personel Per;
cin >> Personelno;

// Girilen kullanıcı bilgisini dosyaya ekle
while ( Personelno > 0 && Personelno <= 100 )
{
// Kullanıcı personel bilgilerini giriyor
cout << "isim, soyisim, maas\n? ";
cin >> adi;
cin >> soyadi;
cin >> maas;

// Girilen personel bilgisi, personel sınıf nesnesi üyelerine atanıyor
Per.setpersonelno( Personelno);
Per.setadi( adi);
Per.setsoyadi( soyadi );
Per.setmaas(maas );
// dosya işaretçisini belirtilen personel no ya konumlandır.
DosyaYaz.seekp( ( Per.getpersonelno() - 1 ) *sizeof( Personel) );
// Personel bilgilerini işaretçinin bulunduğu dosya konumuna kaydet
DosyaYaz.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
// Kullanıcının başka kayıt girmesini sağla
cout << "Personel no gir\n? ";
cin >> Personelno;
} // while döngü sonu
DosyaYaz.close();
}

```

Programda dosya konum işaretçisinin konumu:

```
DosyaYaz.seekp( ( Per.getpersonelno() - 1 ) *sizeof( Personel) );
```

İfadesi ile hesaplanır. Burada dosyada hangi byte a konumlanacağı hesaplanmaktadır. Daha sonrada personel nesnesi için girilen değerler aşağıdaki ifade ile dosyaya kayıt edilir.

```
DosyaYaz.write(reinterpret_cast<const char*>( &Per ),sizeof( Personel ) );
```

17.5.2. 2 Rasgele erişimli dosyadaki kayıtların listelenmesi

Dosyadaki kayıtları listelemek için dosya okuma modunda açılır ve dosya sonuna ulaşana kadar dolu kayıtlar listelenir. Programda bu işlevi aşağıdaki kod parçası gerçekleştirmektedir.

```

if(secim==3)
{
    ifstream DosyaOku("personel.dat",ios::in|ios::binary);

    // Dosya açılmasa çık
    if (!DosyaOku )
    {
        cerr << "Dosya açılmıyor." << endl;
        exit( EXIT_FAILURE );
    } // if bloğu sonu

    // tablo başlığı
    cout << left << setw( 12 ) << "Personelno" << setw( 11 )
    << "adi " << setw( 16 ) << "soyadi" << left
    << setw( 10 ) << right << "maas" << endl;

    Personel Per; // Personel nesnesi oluştur
    DosyaOku.read(reinterpret_cast< char * >( &Per ),sizeof(Personel));
    // dosyadaki kayıtları listele
    while (DosyaOku && !DosyaOku.eof() )
    {
        // okunan kayıtları ekrana yazdır
        if ( Per.getpersonelno() != 0 )
            yazdir( cout, Per );
        DosyaOku.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
    } // while döngü sonu
    DosyaOku.close();
}

```

Burada yazdır fonksiyonu kullanılmıştır. Bu fonksiyon ise

```

// Kayıt yazdır
void yazdir( ostream &buffer, const Personel &kayit )
{
    buffer << left << setw( 12 ) << kayit.getpersonelno()
    << setw( 11 ) << kayit.getadi()
    << setw( 16 ) << kayit.getsoyadi()
    << setw( 10 ) << setprecision( 2 ) << right << fixed
    << showpoint << kayit.getmaas() << endl;
} // yazdir fonksiyonu sonu

```

```

Ne yapmak istiyorsunuz ?
1> dosya oluştur
2> kayıt ekle
3> kayıtları listele
4> Kayıt ara
5> Kayıt düzelt
6> Kayıt sil
3
Personelno   adi           soyadi           maas
1            cemil         oz               30.05
2            ayse         g?l             30.50
27           0           ccc             30.50
Devam etmek için bir tuşa basın . . . _

```

Şekil 17.14 Kayıt listeleme işlevini gösterir ekran çıktısı

17.5.2. 3 Rasgele erişimli dosyada kayıt sorgulama

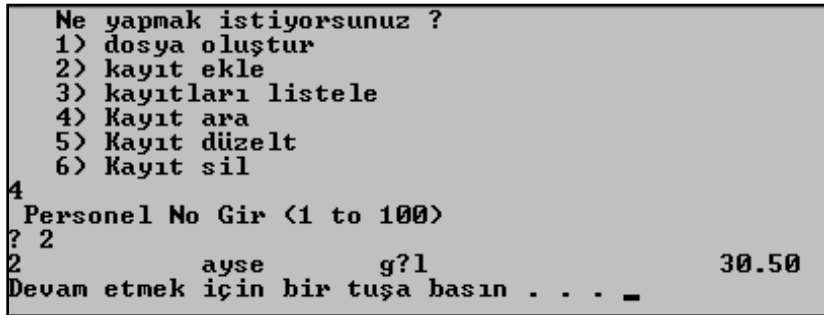
Dosyada belirtilen kayıtları aramak için kayıt eklemede, dosya konum işaretçisi hesaplaması yapıldığı gibi hesaplama yapılır ve daha sonra bu konumdaki kayıt okunarak personel nesnesine atanır.

```
if(secim ==4)
{
    cout << " Personel No Gir (1 to 100)\n? ";

    // Personel no yu gir
    Personel Per;
    cin >> Personelno;
    ifstream DosyaOku("personel.dat",ios::in|ios::binary);

    // Dosya açılmıyor ise çık
    if (!DosyaOku )
    {
        cerr << "dosya açılmıyor" << endl;
        exit( EXIT_FAILURE );
    } // if bloğu sonu

    DosyaOku.seekg( (Personelno - 1)*sizeof(Personel) );
    DosyaOku.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
    yazdir( cout, Per );
    DosyaOku.close();
}
```



```
Ne yapmak istiyorsunuz ?
1> dosya oluşturun
2> kayıt ekle
3> kayıtları listele
4> Kayıt ara
5> Kayıt düzelt
6> Kayıt sil
4
Personel No Gir (1 to 100)
? 2
2          ayse          g?l          30.50
Devam etmek için bir tuşa basın . . . _
```

Şekil 17.15 Kayıt arama işlevini gösterir ekran çıktısı

17.5.2. 4 Rasgele erişimli dosyada kayıt güncelleme

Kayıt düzeltme işlemi için önce belirtilen kayıtların kayıt aramadakine benzer şekilde okunması, daha sonrada nesneye yeni değerlerin atanıp, dosya kayıt işaretçisini aynı konuma getirip dosyaya nesneyi eklemek ile gerçekleştirilir.

```
if(secim ==5)
{
```

```

cout << " Personel No Gir (1 den 100)\n? ";

// personel no belirle
Personel Per;
cin >> Personelno;
fstream Dosya("personel.dat",ios::out|ios::in|ios::binary);

// Dosya açılmıyor ise çık
if (!Dosya )
{
    cerr << "Dosya açılmadı" << endl;
    exit( EXIT_FAILURE );
} // end if

    Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
    Dosya.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
    yazdir( cout, Per );

//nesne üye değişkenlerine yeni değerler ata
cout << "isim, soyisim, maas\n? ";
cin >> adi;
cin >> soyadi;
cin >> maas;

// üye değişken değerlerini ata
Per.setpersonelno( Personelno);
Per.setadi( adi);
Per.setsoyadi( soyadi );
Per.setmaas(maas );
Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
Dosya.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
Dosya.close();
}

```

```

Ne yapmak istiyorsunuz ?
1> dosya oluşturun
2> kayıt ekle
3> kayıtları listele
4> Kayıt ara
5> Kayıt düzelt
6> Kayıt sil
5
Personel No Gir <1 to 100>
? 3
0.00
isim, soyisim, maas
? ali
dal
35.04
Devam etmek için bir tuşa basın . . .

```

Şekil 17.16 Kayıt güncelleme işlevini gösterir ekran çıktısı

17.5.2. 5 Rasgele erişimli dosyada kayıt silme

Dosyadaki bir kayıdı silmek dosyadaki bir kaydın güncellenmesi ile aynıdır. Tek farkı, güncellemede, nesneye güncel bilgilerin girilmesi, silmede ise nesnenin üye değişken değerlerini sıfırlayarak dosyaya kayıt edilmesidir.

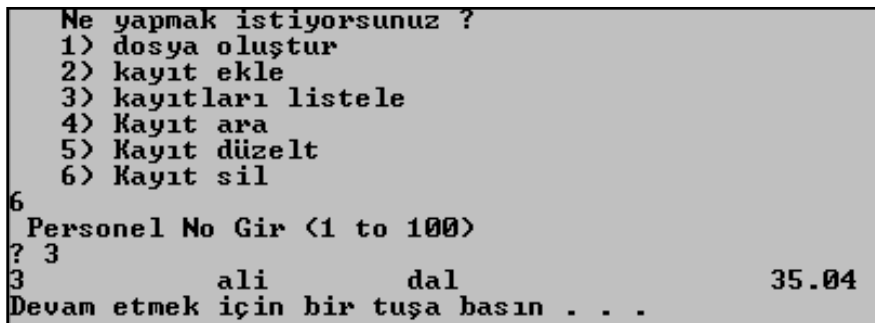
```
if(secim ==6)
{
    cout << " Personel No Gir (1 to 100)\n? ";

    // personel no gir
    Personel Per;
    cin >> Personelno;
    fstream Dosya("personel.dat",ios::out|ios::in|ios::binary);

    // dosya açılmıyor ise çık
    if (!Dosya )
    {
        cerr << "dosya açılmıyor" << endl;
        exit( EXIT_FAILURE );
    } // if bloğu sonu

    Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
    Dosya.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
    yazdir( cout, Per );

    // personel nesnesi değerlerini sıfırla
    Per.setpersonelno( 0);
    Per.setadi( "");
    Per.setsoyadi( "" );
    Per.setmaas(0.00 );
    Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
    Dosya.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
    Dosya.close();
}
```



```
Ne yapmak istiyorsunuz ?
1> dosya oluşturun
2> kayıt ekle
3> kayıtları listele
4> Kayıt ara
5> Kayıt düzelt
6> Kayıt sil
6
Personel No Gir <1 to 100>
? 3
3          ali          dal          35.04
Devam etmek için bir tuşa basın . . .
```

Şekil 17.17 Kayıt silme işlevini gösterir ekran çıktısı.

17.5.2. 6 Rasgele erişimli dosya örneği

Programın tamamı ise aşağıda verilmiştir. Program çalıştırıldığında parça parça açıklanan işlevleri yerine getiren komple bir program elde edilmiş olacaktır.

```
//reinterprint_cast
```

```

//sınıflarda randomdosya örneği(personel) programı
#include <cstdlib>
#include <iostream>
#include <string>
#include <locale.h>
#include <fstream>
#include <iomanip>

using namespace std;

class Personel
{
private:
    int personelNo;
    char soyadi[ 15 ];
    char adi[ 10 ];
    double maas;

public:
// kurucu fonksiyon
    Personel( int = 0, const string & = "", const string & = "", double = 0.0 );

// Personelno için fonksiyonlar
    void setpersonelno( int );
    int getpersonelno () const;

// Personel adı için fonksiyonlar
    void setadi( const string & );
    string getadi () const;

// Personel soyadı için fonksiyonlar
    void setsoyadi( const string & );
    string getsoyadi () const;

// Personel Maas için fonksiyonlar
    void setmaas( double maasD );
    double getmaas() const;

}; //sınıfın sonu

Personel::Personel( int PersonelnoD, const string &adiD, const string &soyadiD, double maasD ) :
personelNo(PersonelnoD),maas(maasD)
{
    setsoyadi(soyadiD );
    setadi( adiD );
} // kurucu fonksiyon sonu

// personelno gir
int Personel::getpersonelno() const
{
    return personelNo;
} // getmaas fonksiyonunun sonu

// Maas değeri
void Personel::setpersonelno( int personelnoD )

```

```

{
personelNo =personelNoD;
} // setmaas fonksiyonunun sonu

// getsoyadi fonksiyonu
string Personel::getsoyadi() const
{
return soyadi;
} // getsoyadi fonksiyonunun sonu

// setsoyadi fonksiyonu
void Personel::setsoyadi( const string &soyadiD )
{
// soyadının en fazla 15 karakter olmasını sağla
int ksay = soyadiD.size();
ksay = ( ksay < 15 ? ksay : 14 );
soyadiD.copy( soyadi, ksay );

soyadi[ ksay ] = '\0'; // soyadının sonuna nul karakteri ekle
} // setsoyadi fonksiyonu sonu

// getadi fonksiyonu
string Personel::getadi() const
{
return adi;
} // getadi fonksiyonu sonu

// setadi fonksiyonu
void Personel::setadi( const string &adiD )
{
// adın en fazla 10 karakter olmasını sağla
int ksay = adiD.size();
adiD.copy(adi,ksay);
ksay = ( ksay < 10 ? ksay : 9 );
adi[ksay ] = '\0'; // null karakteri ekle
} // setadi fonksiyonu sonu

// getmaas fonksiyonu
double Personel::getmaas() const
{
return maas;
} // getmaas fonksiyonu

// setmaas fonksiyonu
void Personel::setmaas( double maasD )
{
maas =maasD;
} // setmaas fonksiyonu sonu

void yazdir( ostream&, const Personel & ); // fonksiyon bildirimi

int main(int argc, char *argv[])
{

```

```

setlocale(LC_ALL,"turkish");

int Personelno;
string adi;
string soyadi;
double maas;

int secim;
do
{
    cout << " Ne yapmak istiyorsunuz ? " << endl;
    cout << " 1) dosya oluřtur " << endl;
    cout << " 2) kayıt ekle " << endl;
    cout << " 3) kayıtları listele " << endl;
    cout << " 4) Kayıt ara " << endl;
    cout << " 5) Kayıt düzelt " << endl;
    cout << " 6) Kayıt sil " << endl;
    cin >> secim;
} while (secim != 1 && secim != 2 && secim != 3 && secim != 4&& secim != 5&& secim != 6);
if(secim==1)
{
    ofstream DosyaYaz( "personel.dat", ios::out | ios::binary );
    // exit program if ofstream could not open file
    if (!DosyaYaz )
    {
        cerr << "dosya açılmıyor" << endl;
        exit(0 );
    } // end if

    Personel Per;

    // dosyaya 100 adet boş kayıt ekle
    for ( int i = 0; i < 100; ++i )
        DosyaYaz.write( reinterpret_cast< const char * >( &Per ),sizeof( Per ) );

}
if(secim==2)
{
    fstream DosyaYaz( "personel.dat", ios::in | ios::out | ios::binary );
    // eęer dosya açılmıyor ise çık
    if (!DosyaYaz )
    {
        cerr << "Dosya açılmıyor." << endl;
        exit( EXIT_FAILURE );
    } // end if
    cout << " Personel No Gir (1 to 100)\n? ";

    // Personel no gir
    Personel Per;
    cin >> Personelno;

    // Girilen kullanıcı bilgisini dosyaya ekle
    while ( Personelno > 0 && Personelno <= 100 )
    {
        // Kullanıcı personel bilgilerini giriyor

```



```

cout << "isim, soyisim, maas\n? ";
cin >> adi;
cin >> soyadi;
cin >> maas;

// Girilen personel bilgisi, personel sınıf nesnesi üyelerine atanıyor
Per.setpersonelno( Personelno);
Per.setadi( adi);
Per.setsoyadi( soyadi );
Per.setmaas(maas );
// dosya işaretçisini belirtilen personel no ya konumlandır.
DosyaYaz.seekp( ( Per.getpersonelno() - 1 ) *sizeof( Personel) );
// Personel bilgilerini işaretçinin bulunduğu dosya konumuna kaydet
DosyaYaz.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
// Kullanıcının başka kayıt girmesini sağla
cout << "Personel no gir\n? ";
cin >> Personelno;
} // while döngü sonu
DosyaYaz.close();
}
if(secim==3)
{
    ifstream DosyaOku("personel.dat",ios::in|ios::binary);

    // Dosya açılmasa çık
    if (!DosyaOku )
    {
        cerr << "Dosya açılmıyor." << endl;
        exit( EXIT_FAILURE );
    } // if bloğu sonu

// tablo başlığı
cout << left << setw( 12 ) << "Personelno" << setw( 11 )
<< "adi " << setw( 16 ) << "soyadi" << left
<< setw( 10 ) << right << "maas" << endl;

Personel Per; // Personel nesnesi oluştur
DosyaOku.read(reinterpret_cast< char * >( &Per ),sizeof(Personel));
// dosyadaki kayıtları listele
while (DosyaOku && !DosyaOku.eof() )
{
    // okunan kayıtları ekrana yazdır
    if ( Per.getpersonelno() != 0 )
        yazdir( cout, Per );
    DosyaOku.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );

} // while döngü sonu
DosyaOku.close();
}
if(secim ==4)
{
    cout << " Personel No Gir (1 to 100)\n? ";

    // Personel no yu gir
    Personel Per;

```

```

cin >> Personelno;
ifstream DosyaOku("personel.dat",ios::in|ios::binary);

// Dosya açılmıyor ise çık
if (!DosyaOku )
{
cerr << "dosya açılmıyor" << endl;
exit( EXIT_FAILURE );
} // if bloğu sonu

    DosyaOku.seekg( (Personelno - 1)*sizeof(Personel) );
    DosyaOku.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
    yazdir( cout, Per );
    DosyaOku.close();
}
if(secim ==5)
{
    cout << " Personel No Gir (1 den 100)\n? ";

    // personel no belirle
    Personel Per;
    cin >> Personelno;
    fstream Dosya("personel.dat",ios::out|ios::in|ios::binary);

    // Dosya açılmıyor ise çık
    if (!Dosya )
    {
        cerr << "Dosya açılmadı" << endl;
        exit( EXIT_FAILURE );
    } // end if

        Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
        Dosya.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
        yazdir( cout, Per );

    cout << "isim, soyisim, maas\n? ";
    cin >> adi;
    cin >> soyadi;
    cin >> maas;

    // üye değişken değerlerini ata
    Per.setpersonelno( Personelno);
    Per.setadi( adi);
    Per.setsoyadi( soyadi );
    Per.setmaas(maas );
    Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
    Dosya.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
    Dosya.close();
}
if(secim ==6)
{
    cout << " Personel No Gir (1 to 100)\n? ";

    // personel no gir

```

```

Personel Per;
cin >> Personelno;
fstream Dosya("personel.dat",ios::out|ios::in|ios::binary);

// dosya açılmıyor ise çık
if (!Dosya )
{
cerr << "dosya açılmıyor" << endl;
exit( EXIT_FAILURE );
} // if bloğu sonu

Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
Dosya.read( reinterpret_cast< char * >( &Per ),sizeof( Personel ) );
yazdir( cout, Per );

// personel nesnesi değerlerini sıfırla
Per.setpersonelno( 0);
Per.setadi( "" );
Per.setsoyadi( "" );
Per.setmaas(0.00 );
Dosya.seekg( (Personelno - 1)*sizeof(Personel) );
Dosya.write( reinterpret_cast< const char * >( &Per ),sizeof( Personel ) );
Dosya.close();
}
system("PAUSE");
return EXIT_SUCCESS;
}
// Kayıt yazdır
void yazdir( ostream &buffer, const Personel &kayit )
{
buffer << left << setw( 12 ) << kayit.getpersonelno()
<< setw( 11 ) << kayit.getadi()
<< setw( 16 ) << kayit.getsoyadi()
<< setw( 10 ) << setprecision( 2 ) << right << fixed
<< showpoint << kayit.getmaas() << endl;
} // yazdır fonksiyonu sonu

```

17.6 Örnek

“dizi.txt” isimli dosyada, 0 ile 4 arasında(0 ve 4 dahil) tamsayı değerler kayıtlıdır. Dosyadan 20 değer okuyan ve bu 20 değer içerisinde şekilde görüldüğü gibi 0 ile 4 arasındaki değerlerin her birinden ne kadar olduğunu hesaplayıp ekrana simgesel olarak yazdıran programını yazınız?

```

//dosyadan okuma örneği
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{

```

```

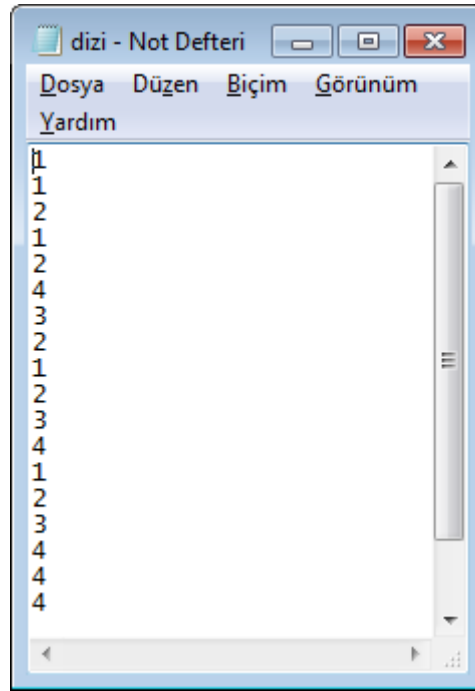
int A[20];
int i,j,ma;
int B[5]={0,0,0,0,0};
//int A[20]={1,1,2,1,2,4,3,2,1,2,3,4,1,2,3,4,4,4,3,1};

ifstream DosyaOku ("dizi.txt");
for(i=0;i<20;i++)
DosyaOku >>A[i];
// Dosyadan okunan deęerler yazdırılıyor
cout<<" dosyadan okunan deęerler"<<endl;
cout<<" =====<<endl;
for(i=0;i<20;i++)
cout<<A[i]<<" ";
cout<<"\n\n";
// frekansını bulma
for (i=0; i<5;i++)
{
    for (j=0; j<20;j++)
    {
        if(i==A[j])
        B[i]++;
    }
}
// en büyük bulma
ma=0;
for (i=0; i<5;i++)
{
    if(ma<B[i])
    ma=B[i];
}

// çizdirme
cout<<" grafiksel gosterim \n\n";
cout<<"\n ";
for(i=1;i<=ma;i++)
cout<<i;
for (i=0; i<5;i++)
{
    cout<<"\n"<<i;
    for( j=1;j<=B[i];j++)
    cout<<"*";
}
cout<<"\n\n";
system("PAUSE");
return EXIT_SUCCESS;
}

```

“dizi.txt” dosyasının içerisindeki veriler şekil 17.18 de,



Şekil 17.18 “dizi.txt” dosyasının içeriği

Programın ekran çıktısı ise şekil 17.19 da verilmiştir.

```
dosyadan okunan de-erler
=====
1,1,2,1,2,4,3,2,1,2,3,4,1,2,3,4,4,4,3,1

grafiksel gosterim

123456
0
1*****
2*****
3*****
4*****
```

Şekil 17.19 örnek programın ekran çıktısı

