BLG221
Veri Yapıları

Giriş

Ders Planı

DERS	KONULAR
01	Giriş, Dosya İşlemleri, Ders örneği: Telefon Rehberi
02	İşaretçiler ve Diziler
03	Fonksiyonlar, Dinamik Bellek Atama, Soyut Veri Tipleri
04	Bağlantılı Listeler
05	Bağlantılı Liste Türleri ve Uygulamaları
06	Yığınlar
07	Kuyruklar
08	Özyinelemeli Programlama
09	Ağaçlar
10	Graflar
11	Hazır Veri Yapısı Kütüphaneleri
12	Algoritma Karmaşıklığı

- Dersin işlenişi:
  - 2 saat konu anlatımı
  - 1 saat uygulama (Asistan)

- Dersin web sayfası: **www.ninova.itu.edu.tr**

  - Tüm duyurular, ödev teslimi vb. sadece dersin web sayfası üzerinden yapılacaktır.
  - Öğrencilerin bu sayfayı düzenli olarak izlemeleri gerekmektedir.

Ders Değerlendirme Kriterleri

Yöntem		Yüzde
Ödevler	(2)	% 10
Proje	(1)	% 10
Vize Sınavı	(1)	% 35
Final Sınavı		% 45

- Geçme notu genelde 40'dır, bazı dönemlerde değişebilir.

Tavsiye Edilen Kaynaklar

1) Data Structures and Algorithm Analysis in C++ (4th edition), Mark Allen Weiss, Addison Wesley, 2014

2) Veri Yapıları ve Algoritmalar (8.basım), Rifat Çölkesen, Papatya Yayıncılık, 2012

3) C How to Program (7th edition), Harvey M. Deitel, Paul J. Deitel, Prentice Hall, 2012

Ön koşullar

- Bil104 veya Bil105 - Introduction to Scientific and Engineering Computing (C) dersini almış olmak, veri yapıları dersinin ön koşuludur.

- **C dilini bilmek** ve küçük örnekler de olsa program yazmış olmak şarttır.
- C bilmeden bu dersi almış öğrencilerin Veri Yapıları'nda başarılı olması mümkün değildir.

Veri Yapıları

- Bilgisayar Bilimlerinde veri yapısı, veriyi etkin bir şekilde kullanabilmek üzere saklama yöntemidir.
- Veri yapısı, verinin matematiksel ve mantıksal organizasyonudur.
- Bir algoritmanın etkin bir şekilde çalışabilmesi için iyi tasarlanmış bir veri yapısı kullanması gerekir.
- İyi tasarlanmış bir veri yapısı, kısıtlı kaynakları (örn: zaman ve bellek) kullanarak birçok kritik işlem yapmayı mümkün kılar.

Veri Yapıları

- Büyük-çaplı yazılım sistemlerinin tasarımında elde edilecek kalite ve performansta, en önemli etkenin, en doğru veri yapısının seçilmesi olduğu görülmüştür.
- Kullanılacak veri yapısı seçildikten sonra, algoritmanın tasarlanması daha kolay bir işlem haline dönüşür.

Veri Yapıları

- Dizi
- Bağlantılı Liste
- Yığın
- Kuyruk
- Ağaç
- Graf
- ...

Temel İşlemler

- Ekleme/Silme/Güncelleme
- Arama
- Sıralama
- Dolaşma
- ...

Bu veri yapılarının kendi içinde alt türleri de vardır.

Algoritma Türleri

- **Döngüsel (iterative)**
  - Sadece döngü komutları kullanılır (for, while, do)

- **Özyinelemeli (recursive)**
  - Kendi kendisini çağıran fonksiyonlardır.
  - İki kısımdan oluşur:
    - Sonlandırma komutları (base case)
    - Rekürsif çağırma komutları (recursive case)

Veri Yapılarının Gerçekleştirilmesi (Implementation)

- Dizi (array) en basit veri yapısıdır, fakat çoğu kez etkin olmayan bir yöntemdir.
- ***Bağlantılı Liste (Linked List) en temel ve yaygın kullanılan veri yapısıdır.***
- Tüm diğer soyut veri yapısı türleri, ya Dizi kullanarak veya Bağlantılı Liste kullanarak gerçekleştirilebilir.

Veri Yapıları nerelerde kullanılır ?

- Bazı örnekler:

Kullanım Alanı	Veri Yapıları
Ağ Yönetimi ve İletişim	Graf, Kuyruk
Bilgi Erişimi	Ağaç, Liste
İşletim Sistemi	Yığın, Kuyruk, Liste
Veri Tabanı Yönetim Sistemi	Ağaç, Liste
Derleyici	Yığın, Liste, Ağaç
Bilgisayar Grafiği , Görüntü İşleme	Liste
Olay-tabanlı Simülasyon	Kuyruk
Kaynak Atama , Optimizasyon	Kuyruk
Durum Uzayı Araması (Yapay Zeka)	Ağaç

Ders Örnekleri

- Örnek kodlarda *C* programlama dili ve yapısal programlama kullanılacaktır.
- Yeni standartlara uyumluluk açısından kodlarda, *C++* dilinin getirdiği bazı yeniliklerden faydalanılacaktır.
- Bu özellikler kullanıldıkları yerlerde belirtilecektir.

Örnek:

C

```
printf("%d\n", sayi);
scanf("%d", &sayi);
```

C++

```
cout << sayi << endl;
cin >> sayi;
```

C++ ile C arasındaki bazı farklar

- **Veri Yapıları dersinde aşağıdaki C++ özellikleri kullanılacaktır.**

C	C++
#include <stdio.h> printf("%d. sıradaki verileri girin : ", i); scanf("%d %f", &UrunNum, &Fiyat)	#include <iostream> using namespace std; cout << i << ". sıradaki verileri girin : "; cin >> UrunNum >> Fiyat;

int * a;
a = (int *) malloc(sizeof(int));
free(a);
	int * a;
	a = new int;
	delete a;

struct Ornek {
int x1, int x2;
}
	struct Ornek {
	int x1, int x2;
	float ort_hesapla()
	{ return (x1+x2) /2.0;}
	}

#define TRUE 1 #define FALSE 0 int a; a = TRUE;	bool a; a = true;

C++'da std nesnesi (namespace)

- Standart C++ header dosyalarında, tüm deklarasyonlar **std** denilen bir isim uzayında (namespace) tanımlıdırlar.

```
#include <iostream>
using namespace std;
```

- C++'da ekrana yazma ve klavyeden okuma için **cin** ve **cout** nesneleri kullanılır. C'deki karşılıkları **printf** ve **scanf** fonksiyonlarıdır.

- **iostream** isimli header dosyası aşağıdaki nesneleri içerir:

  - **cin** : Standard veri giriş nesnesidir (klavye).
    Okunacaklar listesi **"input stream extraction operator"** (>>) ile ayrılırlar.

  - **cout** : Standard veri çıkış nesnesidir (ekran).
    Yazılacaklar listesi **"output stream insertion operator"** (<<) ile ayrılırlar.

Örnekler

- Program başlangıcında **using namespace std;** komutu yazılmazsa, cin ve cout nesnelerinin her kullanımında **std::** ön eki yazılmak zorundadır. Aksi halde derleyici hata verir.
- **:: işareti scope resolution operatörüdür.**

Program1

```
#include<iostream>

int main() {
   std::cout << "Hello" << endl;
   return 0;
}
```

Program2

```
#include<iostream>
using namespace std;

int main() {
   cout << "Hello" << endl;
   return 0;
}
```

C dilinde << ve >> operatörlerinin diğer kullanım amaçları

Standard C'de bu operatörler, bitwise shift işlemi yapmak üzere aritmetik ifadelerde kullanılabilir. Bit kaydırma işlemi iki operand gerektirir.

Operatör	Anlamı	Açıklama
<<	Bitwise left-shift operator (sola kaydırma)	En soldaki bit kaybolur, en sağdaki bit sıfırlanır.
>>	Bitwise right-shift operator (sağa kaydırma)	En sağdaki bit kaybolur, en soldaki bit sıfırlanır.

```
int main() {
  int a = 5 << 2;    // Shift bits of 5 left 2 times.
  int b = 20 >> 2;  // Shift bits of 20 right 2 times.
  cout << a << endl;
  cout << b << endl;
  return 0;
}
```

Ekran çıktısı:

```
20
5
```

Derleyici Önişlem Direktifleri

- Önişlemler
  - Direktif satırları # işareti ile başlar
  - Program derlenmeden önce ön işlemler yapılır

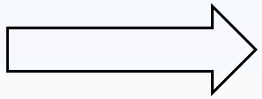
- Kullanım yerleri
  - Kaynak dosyaların (header, vb.) include edilmesi
  - Sembolik sabitler ve makrolar tanımlanması
  - Koşullu derleme satırları

`#define`	`#undef`	`#ifdef`	`#ifndef`
`#if`	`#else`	`#endif`	`#elif`
`#include`	`#error`	`#line`	`#pragma`

Koşullu Derleme

- Derleyicinin preprocessor direktiflerini kontrol eder.
- Sentaksı if deyimine benzerdir.
- Örnek: Sembolik sabit PI'nin daha önce tanımlanıp tanımlanmadığını test edelim.
  - Eğer PI tanımlı ise, **defined(PI)** testinin sonucu 1'dir
  - Eğer PI tanımlı değilse, değerini 3.14 yapar
- Her bir #if direktifine karşılık gelen bir #endif yazılmalıdır.

```
#if !defined( PI )
   #define PI 3.14
#endif
```



```
#ifndef  PI
   #define PI 3.14
#endif
```

Aynıdırlar

Ders Örneğinin Tanıtımı

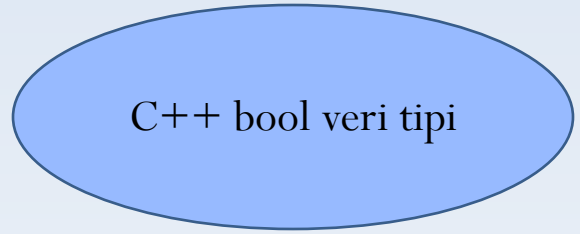
TELEFON DEFTERİ

Kayıtlı kişilerin numaralarını gösteren; kayıt ekleme, kayıt silme, kayıt güncelleme, kayıt arama özellikleri bulunan bir telefon defteri uygulaması gerçeklenecektir.

C kodlarına Ninova web sitesi üzerinden erişilebilir.

Telefon Defteri

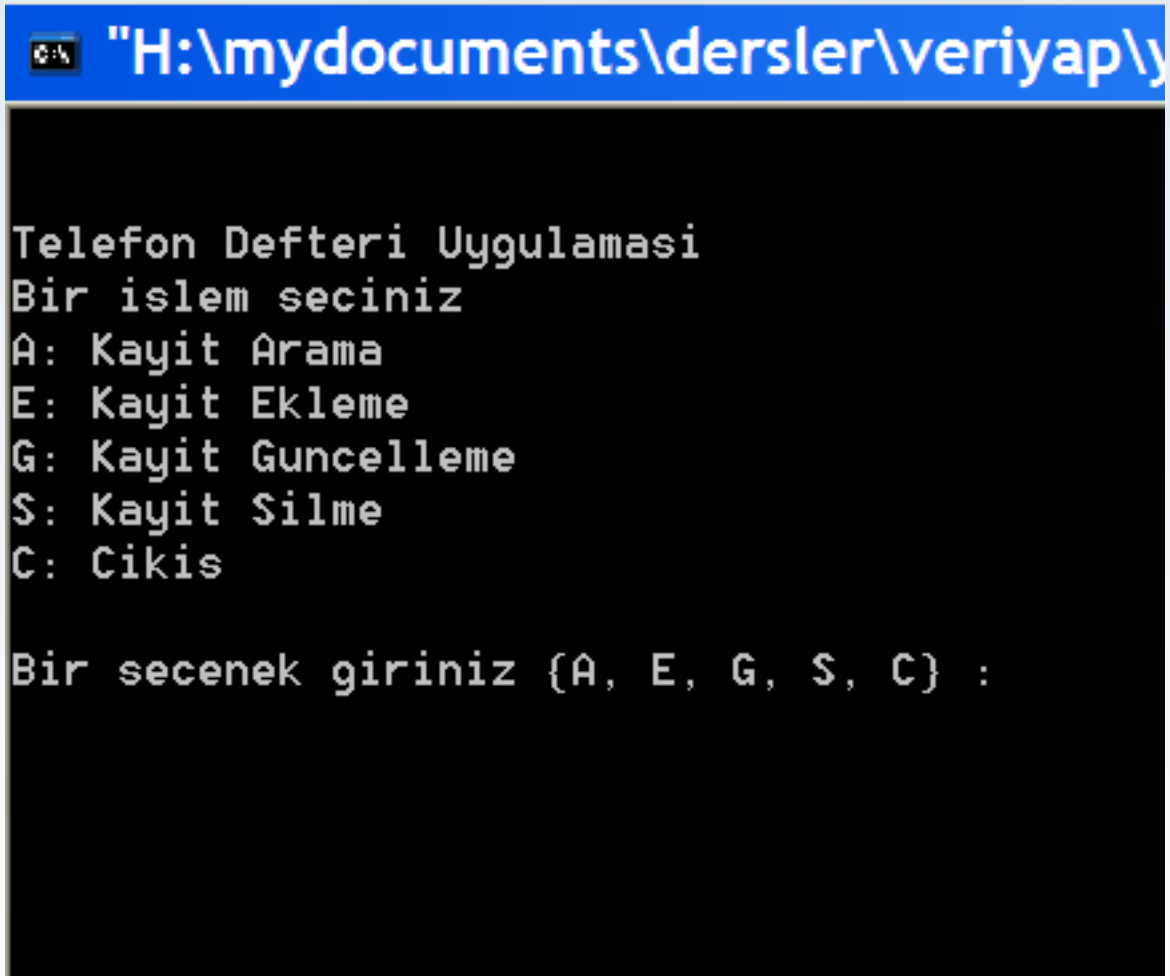
```
int main(){
  bool bitir = false;
  char secim;
  while (!bitir) {
      menu_yazdir();
      cin >> secim;
      bitir = islem_yap(secim);
  }
  return EXIT_SUCCESS; // return 0
}
```



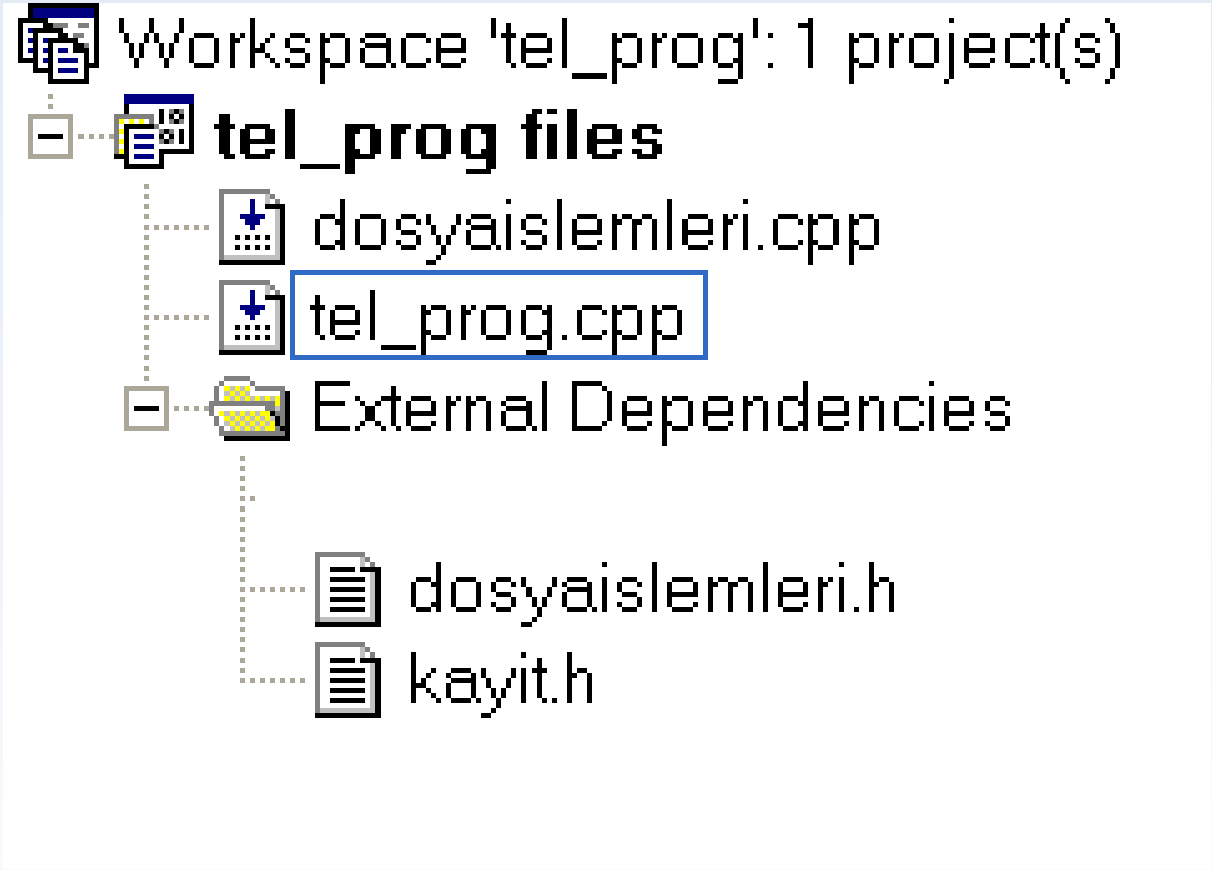
menu_yazdır() Fonksiyonu

```
void menu_yazdir(){
  system("clear");
  cout << endl << endl;
  cout << "Telefon Defteri Uygulamasi" << endl;
  cout << "Bir islem seciniz" << endl;
  cout << "A: Kayit Arama" << endl;
  cout << "E: Kayit Ekleme" << endl;
  cout << "G: Kayit Guncelleme" << endl;
  cout << "S: Kayit Silme" << endl;
  cout << "C: Cikis" << endl;
  cout << endl;
  cout << "Bir secenek giriniz {A,E,G,S,C}: “;
}
```

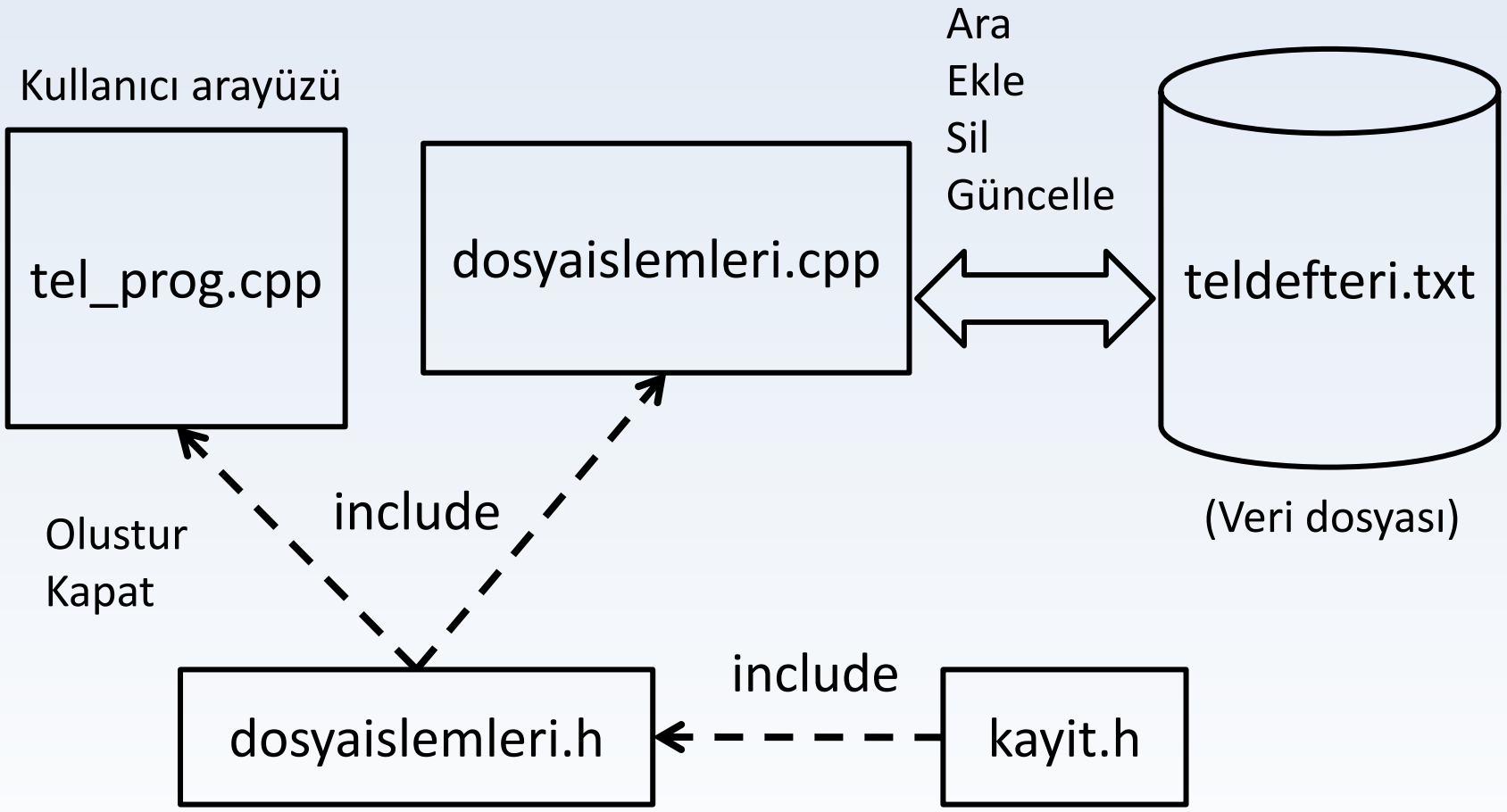
menu_yazdir


"H:\mydocuments\dersler\veriyap\y
Telefon Defteri Uygulamasi
Bir islem seciniz
A: Kayit Arama
E: Kayit Ekleme
G: Kayit Guncelleme
S: Kayit Silme
C: Cikis
Bir secenek giriniz {A, E, G, S, C} :

Projedeki Kaynak Dosyalar



Projedeki Kaynak Dosyalar



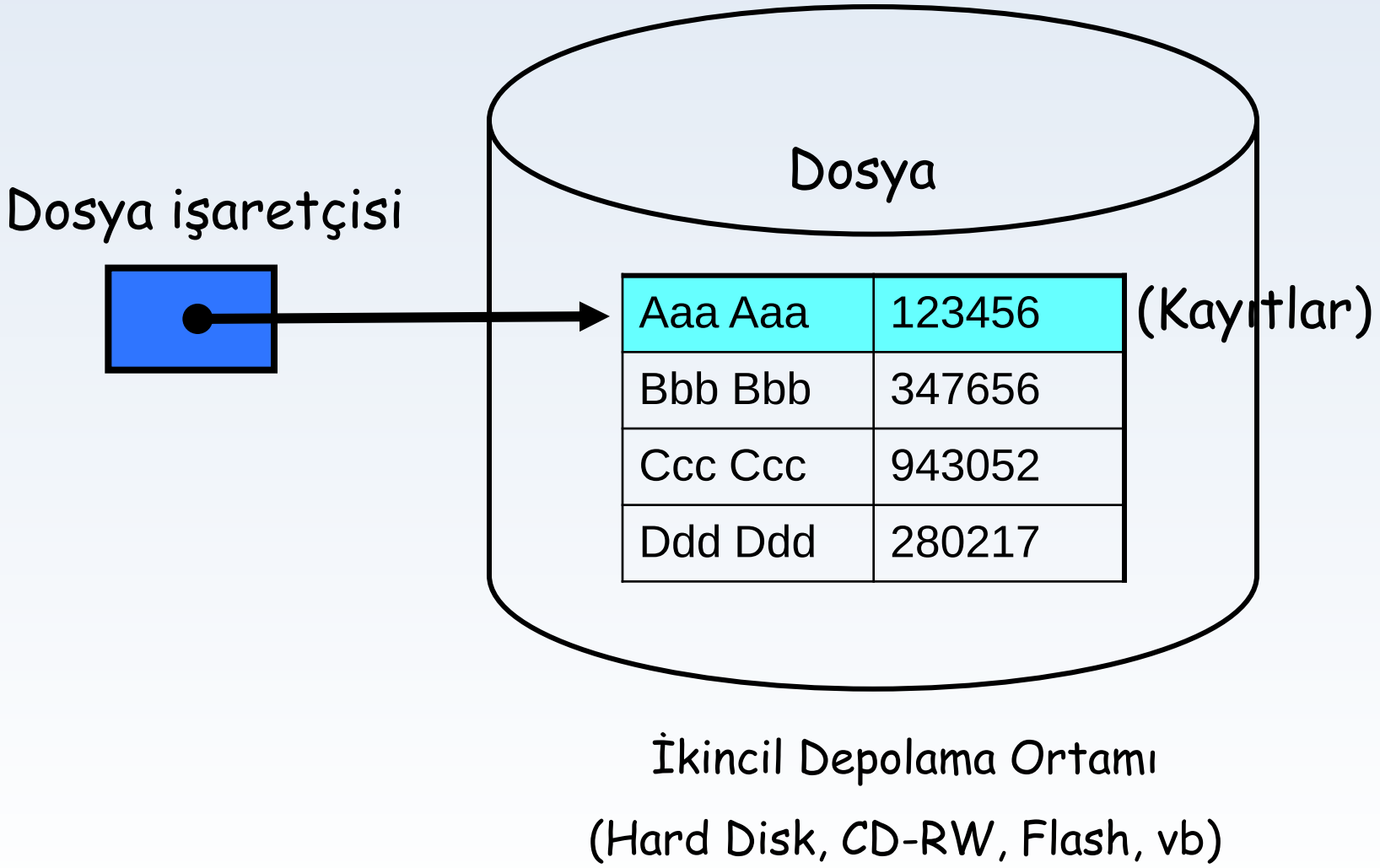
Veri Dosyası yapısı

- Bellekteki (RAM) yapılarda tutulan veriler geçicidir ve programın kapanmasıyla birlikte yok olurlar.
- Dosyalar ise verileri kalıcı olarak saklayabileceğimiz bir ortam sağlarlar.
- Bilgisayarlar dosyaları ikincil depolama birimlerinde (sabit diskler, manyetik diskler, optik diskler ...) saklarlar.
- İkincil depolama birimlerinde işlem yapmak bellekte işlem yapmaya göre daha yavaştır.

Kayıt yapısı (kayit.h)

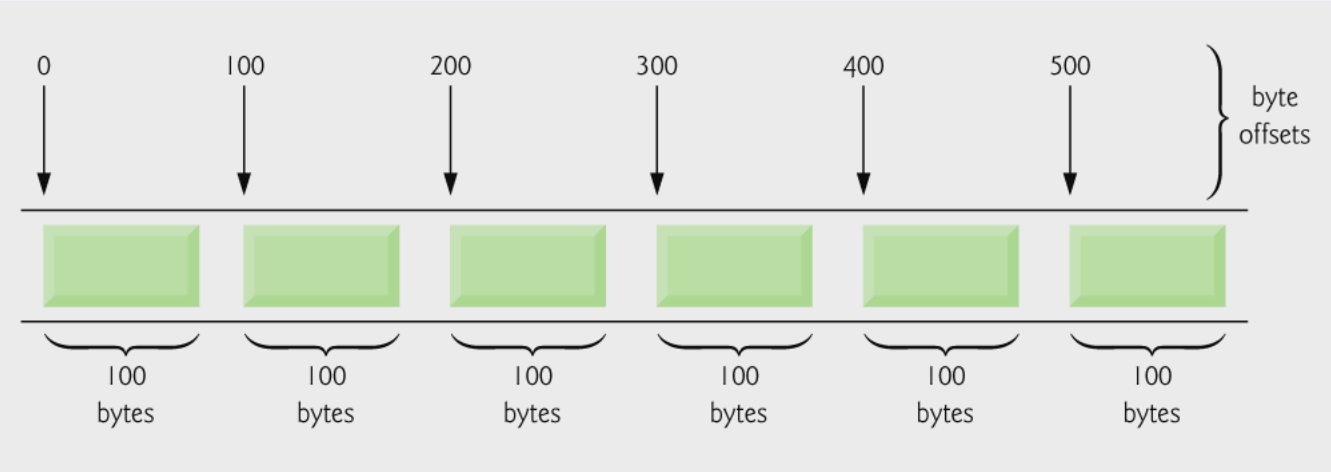
```
#define AD_UZUNLUK 30
#define TELNO_UZUNLUK 15

struct Tel_Kayit{
  char ad[AD_UZUNLUK];
  char telno[TELNO_UZUNLUK];
};
```

Dosya ve Kayıtların Mantıksal Gösterimi (Telefon defteri örneği)



Rastgele Erişimli Dosyada Kayıtlar

- Rastgele erişilebilir dosyalar
  - Kayıtlara doğrudan erişim imkanı sağlarlar
  - Sabit boyutlu kayıtlar üzerinde çalışırlar
  - Sequential dosyalardan farkı, kayıt güncellemeye imkan sağlamasıdır.



Rastgele Erişimli Dosya Fonksiyonları

- Aşağıdaki fonksiyonlar **<stdio.h>** header dosyasında tanımlıdır.
- **fseek:** Dosya konumunu belirli bir lokasyona getirir. (Dosyada bazı kayıtları atlamak için kullanılır.)
- **fread:** Veri dosyasından byte'ları, bellekteki bir değişkene transfer eder. (Formatsız giriş.)
- **fwrite:** Bellekteki bir değişkendeki byte'ları, bir veri dosyasına transfer eder. (Formatsız çıkış.)

fseek Foknsiyonu

```
fseek(filePtr, offset, symbolic_constant );
```

- ***filePtr*** : Dosya işaretçisi
- ***offset*** : Dosya konum mesafesi (ilk kayıt 0'dan başlar)

- ***symbolic_constant*** : Göreceli konum belirteci
  - SEEK_SET : dosya başından itibaren
  - SEEK_CUR : dosyada en son bulunulan konumdan itibaren
  - SEEK_END : dosya sonundan itibaren

fwrite ve fread Fonksiyonları

`fwrite( &variable_name,`
`sizeof(variable),`
`number_of_blocks,`
`filePtr );`
	`fread( &variable_name,`
	`sizeof(variable),`
	`number_of_blocks,`
	`filePtr );`

- **&variable_name** : Bellekte transfer yapılacak yer.
- **sizeof(variable)** : Transfer edilecek byte sayısı.
- **number_of_ blocks** : Diziler için kaç adet elemanın transfer edileceğini belirtir. Genellkle her defasında 1 eleman transfer edilir.
- **filePtr** : Dosya işaretçisi.

C++ ve Struct

- C++ dilinin struct yapısı, soyut veri tiplerini tanımlamada doğal bir kapsül oluşturur.
- Böylece veri tipi ve bu tipin üzerinde yapılacak işlemleri tanımlayan fonksiyonlar, aynı kapsülün içinde yeralırlar ve mantıksal olarak birbirlerine bağlanırlar.
- C'den farklı olarak C++'da struct içinde fonksiyonlar da yazılabilir.
- C++'da struct ile class tanımları hemen hemen aynıdır.

C

```
typedef struct VeriTipi{
   int dizi[10];
   int elemansayisi;
} YeniDiziTipi;

int KacEleman(YeniDiziTipi *d){
   return d->elemansayisi;
}
```

C++

```
struct VeriTipi{
   int dizi[10];
   int elemansayisi;
   int KacEleman ();
};

int VeriTipi::KacEleman(){
   return elemansayisi;
}
```

telprog.cpp

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
#include <conio.h>
#include <ctype.h>
#include "dosyaislemleri.h"

using namespace std;

typedef Dosya Veriyapisi;
Veriyapisi defter;

void menu_yazdir();
bool islem_yap(char);
void kayit_ara();
void kayit_ekle();
void kayit_sil();
void kayit_guncelle();

int main(){ ...
```

main() Program

```
typedef Dosya Veriyapisi;
Veriyapisi defter;

int main(){
  defter.olustur();
  bool bitir = false;
  char secim;
  while (!bitir) {
      menu_yazdir();
      cin >> secim;
      bitir = islem_yap(secim);
      }
  defter.kapat();
  return EXIT_SUCCESS;
}
```

dosyaislemleri.h

```
#ifndef DOSYAISLEMLERI_H
#define DOSYAISLEMLERI_H
#include <stdio.h>
#include "kayit.h"

struct Dosya{
  char *dosyaadi;
  FILE *teldefteri;
  olustur();
  kapat();
  ekle(Tel_Kayit *);
  int ara(char []);
  sil(int kayitno);
  guncelle(int kayitno, Tel_Kayit *);
};
#endif
```

dosyaislemleri.cpp

```
#include "dosyaislemleri.h"
#include <iostream>
#include <stdlib.h>
#include <cstdlib.h>
#include <string.h>

using namespace std;

Dosya::olustur(){ ...}
...
...
```

Dosya::olustur() Fonksiyonu

```
Dosya::olustur(){
  dosyaadi="teldefteri.txt";
  teldefteri = fopen( dosyaadi, "r+" );
  if(!teldefteri){
     if(!(teldefteri=fopen(dosyaadi,"w+"))){
          cerr << "Dosya acilamadi" << endl;
          exit(1);
     }
  }
}
```

Dosya::kapat() Fonksiyonu

```
Dosya::kapat(){
  fclose(teldefteri);
}
```

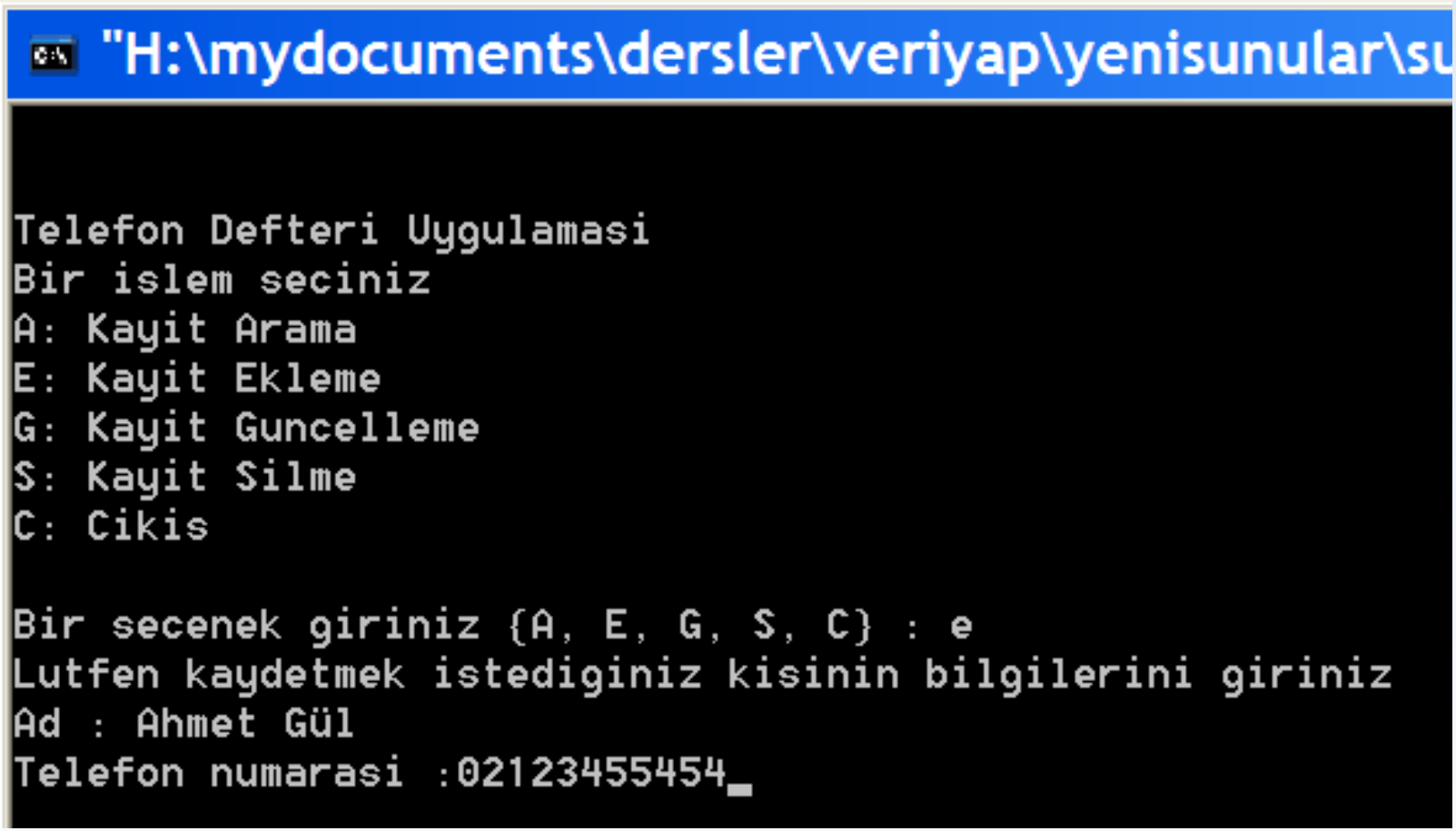
islem_yap() Fonsiyonu

```
bool islem_yap(char secim){
   bool sonlandir=false;
   switch (secim) {
          case 'A': case 'a':
                    kayit_ara();  break;
          case 'E': case 'e':
                    kayit_ekle(); break;
          case 'G': case 'g':
                    kayit_guncelle(); break;
          case 'S': case 's':
                    kayit_sil(); break;
          case 'C': case 'c':
                    cout << "Programı sonlandirmak istediğinize emin misiniz? (E/H):";
                    cin >> secim;
                    if(secim=='E' || secim=='e') sonlandir=true; break;
          default:
                    cout << "Hata: Yanlis giris yaptiniz" << endl;
                    cout << "Tekrar deneyiniz {A, E, G, S, C} :" ;
                    cin >> secim;
                    sonlandir = islem_yap(secim);
                    break;
   }
   return sonlandir;
}
```

kayit_ekle() Fonksiyonu

```
void kayit_ekle(){
  Tel_Kayit yenikayit;
  cout << "Lutfen kaydetmek istediginiz kisinin
      bilgilerini giriniz" << endl;
  cout << "Ad : " ;
  cin.getline(yenikayit.ad,AD_UZUNLUK);
  cout << "Telefon numarasi :";
  cin >> setw(TELNO_UZUNLUK) >>yenikayit.telno;
  defter.ekle(&yenikayit);
  cout << "Kaydiniz eklenmistir" << endl;
  getchar();
};
```

kayit_ekle


"H:\mydocuments\dersler\veriyap\yenisunular\su
Telefon Defteri Uygulamasi
Bir islem seciniz
A: Kayit Arama
E: Kayit Ekleme
G: Kayit Guncelleme
S: Kayit Silme
C: Cikis
Bir secenek giriniz {A, E, G, S, C} : e
Lutfen kaydetmek istediginiz kisinin bilgilerini giriniz
Ad : Ahmet Gül
Telefon numarasi :02123455454

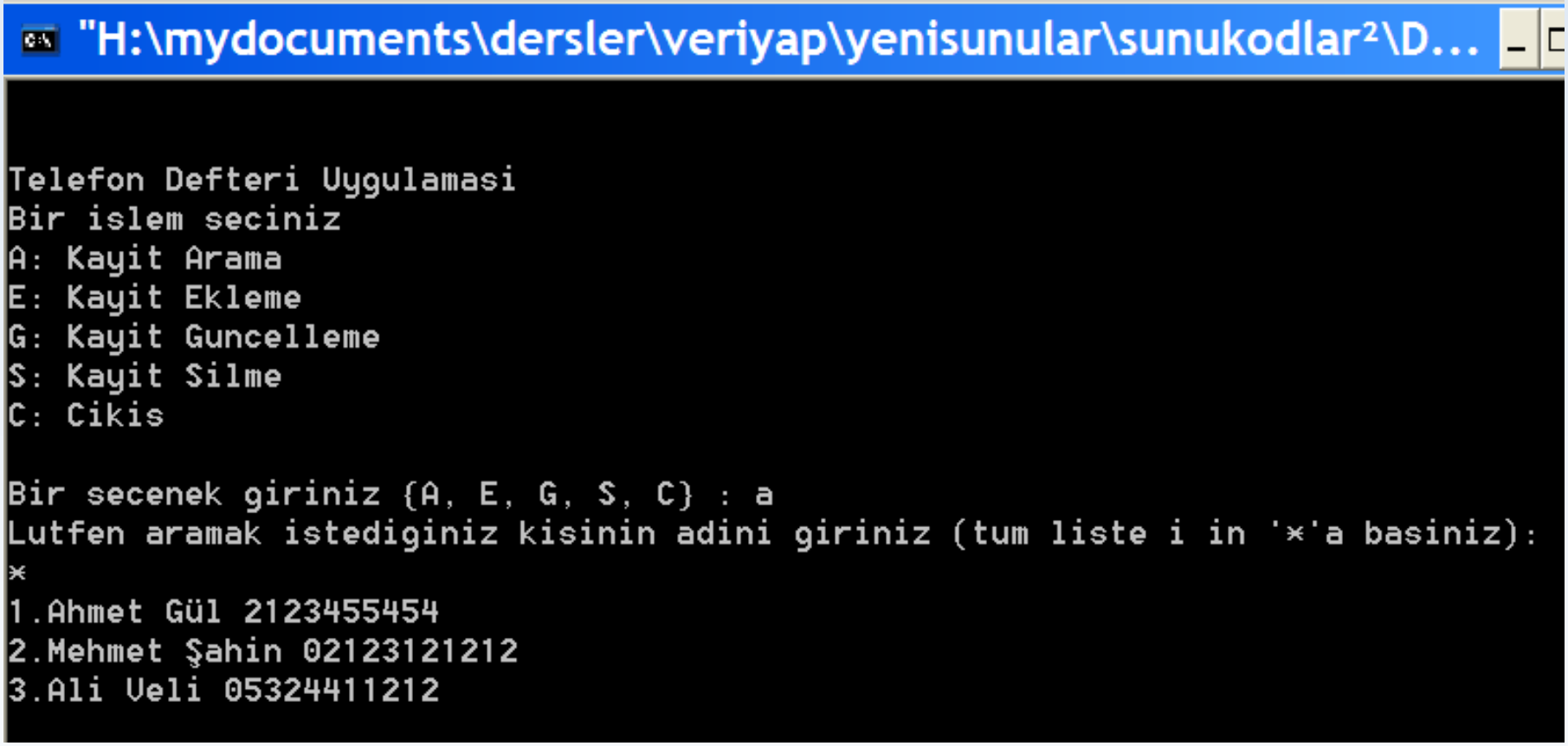
Dosya::ekle() Fonksiyonu

```
Dosya::ekle(Tel_Kayit *ykptr){
  fseek(teldefteri, 0, SEEK_END);
  fwrite(ykptr, sizeof(Tel_Kayit), 1, teldefteri);
}
```

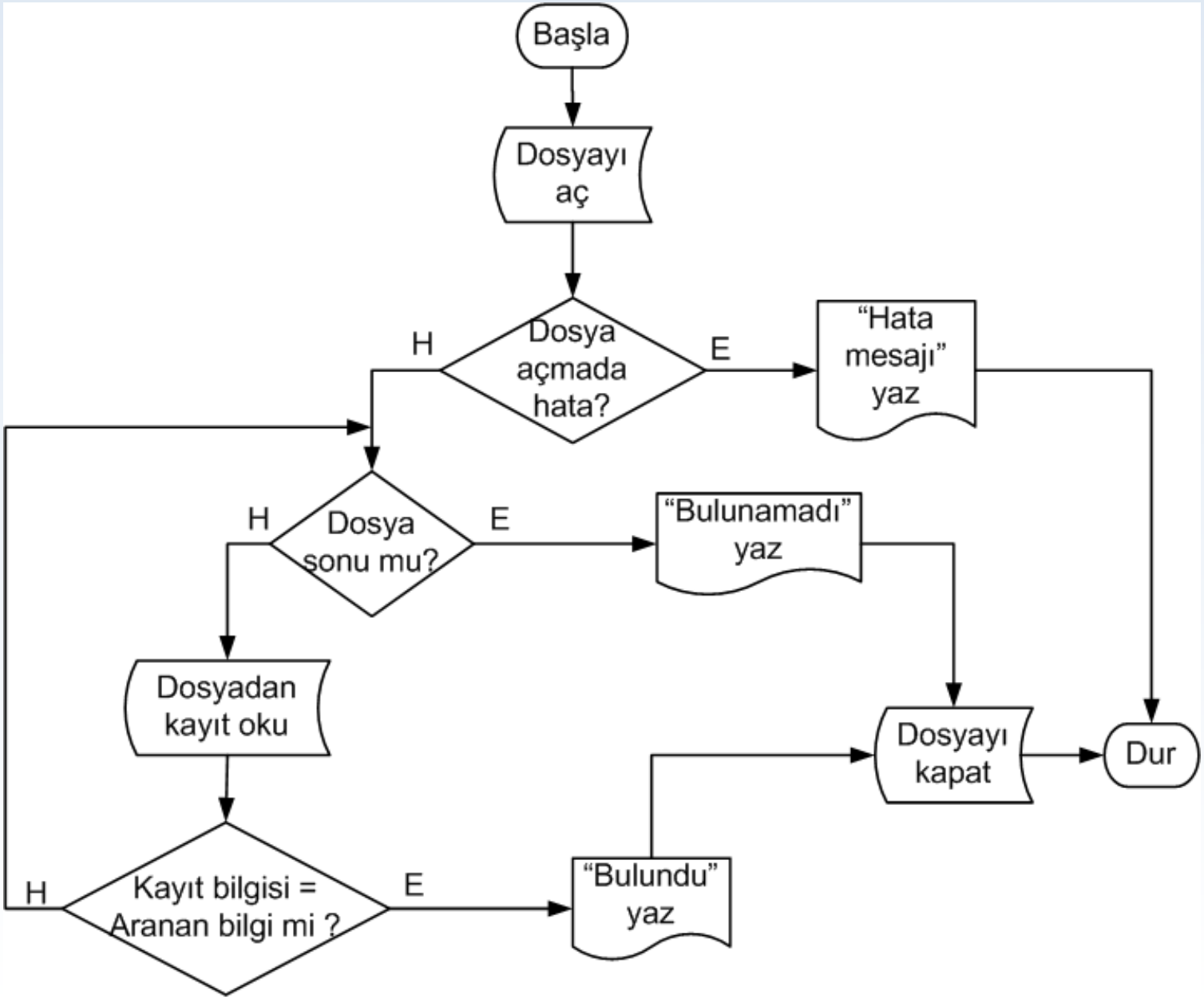
kayit_ara() Fonksiyonu

```
void kayit_ara(){
  char ad[AD_UZUNLUK];
  cout << "Lutfen aramak istediginiz kisinin
  adini giriniz (tum liste için '*'a basiniz):"
  << endl;
  cin.getline(ad,AD_UZUNLUK);
  if(defter.ara(ad)==0){
      cout << "Aradiginiz kriterlere uygun kayit
  bulunamadi" << endl;
  }
  getchar();
};
```

kayit_ara


"H:\mydocuments\dersler\veriyap\yenisunular\sunukodlar²\D...
Telefon Defteri Uygulamasi
Bir islem seciniz
A: Kayit Arama
E: Kayit Ekleme
G: Kayit Guncelleme
S: Kayit Silme
C: Cikis
Bir secenek giriniz {A, E, G, S, C} : a
Lutfen aramak istediginiz kisinin adini giriniz (tum liste i in '*'a basiniz):
*
1.Ahmet Gül 2123455454
2.Mehmet Şahin 02123121212
3.Ali Veli 05324411212

Dosyada kayıt arama için genel algoritma



Dosya::ara() Fonksiyonu

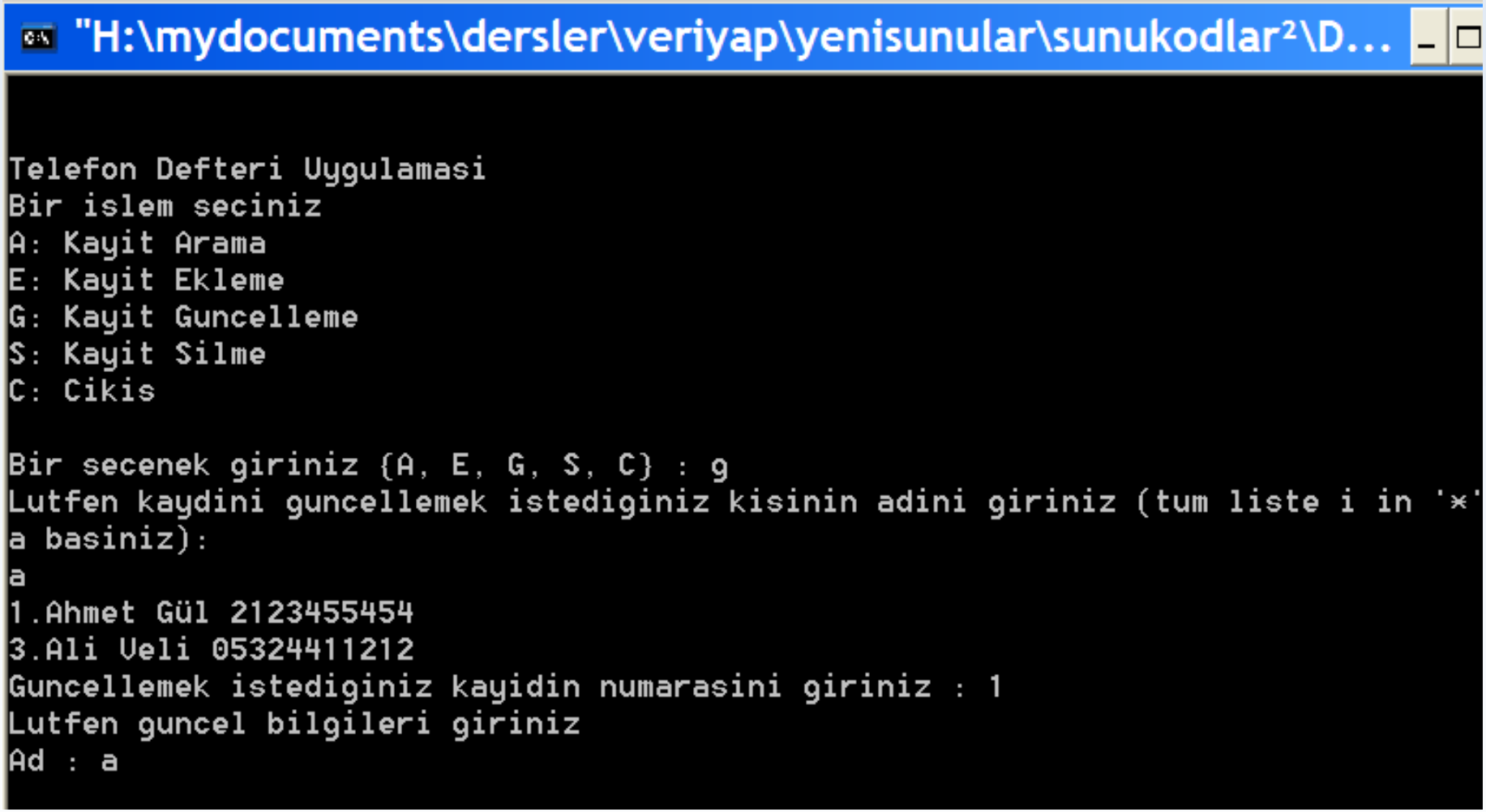
```
int Dosya::ara(char *aranacak){
  Tel_Kayit k;
  int sayac=0;
  bool tumu=false;
  int bulunan=0;
  if(strcmp(aranacak,"*")==0)
       tumu=true;
  fseek(teldefteri, 0, SEEK_SET);
  while(!feof(teldefteri)){
       sayac++;
       fread( &k, sizeof (Tel_Kayit), 1, teldefteri);
       if(feof(teldefteri)) break;
       if(tumu || strnicmp(k.ad,aranacak,strlen(aranacak))==0){
              cout <<sayac <<"."<< k.ad <<" "<< k.telno <<endl;
              bulunan++;
       }
  }
  return bulunan;
}
```

kayit_güncelle() Fonksiyonu

```
void kayit_guncelle(){
  char ad[AD_UZUNLUK];
  int secim;
  cout << "Lutfen kaydini guncellemek istediginiz kisinin adini
       giriniz (tum liste için '*'a basiniz):" << endl;
  cin.getline(ad,AD_UZUNLUK);
  int kisisayisi=defter.ara(ad);
  if(kisisayisi==0){
       cout << "Aradiginiz kriterlere uygun kayit bulunamadi" << endl;
  }
  else {
        if (kisisayisi==1){
              cout << "Kayit bulundu." << endl;
              cout << "Eger guncellemek istediginiz kayit buysa lutfen
              numarasini giriniz (Islem yapmadan cikmak icin -1
              giriniz): " ;
       }else cout << "Guncellemek istediginiz kayidin numarasini
              giriniz (Islem yapmadan cikmak icin -1 giriniz): " ;
```

```
        cin >> secim;
        if(secim==-1) return;
        Tel_Kayit yenikayit;
        cout << "Lutfen guncel bilgileri giriniz" <<
              endl;
        cout << "Ad : ";
        cin.getline(yenikayit.ad,AD_UZUNLUK);
        cout << "Telefon numarasi :";
        cin >> setw(TELNO_UZUNLUK) >> yenikayit.telno;
        defter.guncelle(secim,&yenikayit);
        cout << "Kaydiniz basariyla guncellendi"
              <<endl;
  }
  getchar();
};
```

kayit_güncelle



Dosya::güncelle() Fonksiyonu

```
Dosya::guncelle(int kayitno,Tel_Kayit *ykptr)
{
  if(fseek(teldefteri,
           sizeof(Tel_Kayit)*(kayitno-1),
           SEEK_SET) == 0)
     fwrite(ykptr,
            sizeof(Tel_Kayit), 1, teldefteri);
}
```

kayit_sil() Fonksiyonu

```
void kayit_sil(){
  char ad[AD_UZUNLUK];
  int secim;
  cout << "Lutfen kaydini silmek istediginiz kisinin
       adini giriniz (tum liste için '*'a basiniz):" << endl;
  cin.getline(ad,AD_UZUNLUK);
  int kisisayisi=defter.ara(ad);
  if(kisisayisi==0){
       cout << "Aradiginiz kriterlere uygun kayit
       bulunamadi" << endl;
  }
```

```
  else {
     if (kisisayisi==1){
       cout << "Kayit bulundu." << endl;
       cout << "Eger silmek istediginiz kayit buysa
             lutfen numarasini giriniz (Islem yapmadan
             cikmak icin -1 giriniz): " ;
    }
    else cout << "Silmek istediginiz kayidin
       numarasini giriniz (Islem yapmadan cikmak
       icin -1 giriniz): " ;
    cin >> secim;
    if(secim==-1) return;
    defter.sil(secim);
    cout << "Kayit silindi" <<endl;
  }
  getchar();
};
```

Dosya::sil() Fonksiyonu

```
Dosya::sil(int kayitno)
{
 Tel_Kayit boskayit={"",""};
 if(fseek(teldefteri,
         sizeof(Tel_Kayit)*(kayitno-1),
          SEEK_SET)==0)
     fwrite(&boskayit,
             sizeof(Tel_Kayit),
             1,teldefteri);
}
```

Uygulamada yapılacaklar

1. Program derleme aşamaları ve işlemlerinin anlatılması.
2. Ders örneği üzerinde gerçek silme işleminin gerçeklenmesi.
3. Ders örneği üzerinde alfabetik sıraya göre listeleme işleminin (indis dizisi kullanarak) gerçeklenmesi.