İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

VERİ TABANI YÖNETİM SİSTEMİ

YÜKSEK LİSANS TEZİ

Müh.İclal YILDIRIM

Tezin Enstitüye Verildiği Tarih : 4 Haziran 1990
Tezin Savunulduğu Tarih :22 Haziran 1990

Tez Danışmanı : Doç.Dr.Mithat UYSAL
Diğer Jüri Üyeleri : Prof.Dr.Galip G.TEPEHAN
Doç.Dr.Fikret BALTA

Haziran - 1990

ÖNSÖZ

Veri tabanı yönetim sistemleri, geçmişte yüksek bellek kapasiteli ve hızlı işleyebilen bilgisayar aracılığıyla uygulanabilirlerdi. Bu günlerde geliştirelen bilgisayarlar aracılığı ile düşük fiyatlı olarak az bir yatırımla ve etkin olarak gerçekleştirilebiliyorlar. Daha fazla veri tabanı yönetim sistemi, kişisel ve iş dünyası için geliştirilebiliyor ve kullanılabiliyor.

Böyle bir konuda, bana yüksek lisans tezi vererek kendileri ile çalışma imkanı sağlamış olan Doç.Dr.Mithat UYSAL'a, bilgisayar konusundaki sorunlarıma yardımcı olan bölüm başkanım Prof.Dr.Belgin MAZLUMOĞLU'na, tez çalışmalarımı yürüttüğüm süre zarfında kızım Özlemin bakımına yardımcı olarak beni destekleyen annem, babam ve eşime, yazılım aşamasındaki yardımlarından dolayı Şahinaz YÜZBAŞIOĞLU'na şükranlarımı arz ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÖZET

Veri tabanı, büyük veri topluluklarının yönetilmesi ve düzenlenmesi için kullanılan, sistematik bir yaklaşımdır. Bir veri tabanı yönetim sisteminde yalnız veriler değil, veriler arasındaki bir takım ilişkilerde saklanır.

Veri tabanı yönetim sisteminin önemli işlevleri, veri tabanı içindeki herhangi bir bilginin yerinin bulunması ve ona erişilmesi, veri kayıtlarının istenilen sırada sıralanması ve indeks yaratmaktır.

Bu çalışmada veri kayıtlarının girişi, aranması ve silinmesi için lineer hash adresleme metodu kullanılmıştır.

Lineer hash adresleme metodu çok yeni bir metoddur ve henüz birçok sistemde kullanılmamıştır.

Dosyaya ne kadar kayıt eklenirse eklensin, yükleme faktörü sabit tutularak bellek kullanımı ve kayıtlara tek tek erişme açısından etkin bir metod olduğu gözlenmiştir. Bu metodla Anahtar alanların birbirlerine bağlanması bağlı liste yapısı kullanılarak gerçekleştirilmiştir.

Lineer hash metodu anlatılırken B^+ ağaçları ve diğer metodlarla mukayese edilmiştir.

Veri alanlarının sıralanmasında ise quick-sort yöntemi kullanılmıştır.

SUMMARY

DATA BASE MANAGEMENT SYSTEM

This work a program that helps you collect and manage information or data

What is a database:

A database is organized collection of related information or data. You probably encounter several databases every day. You take care of your personal business with database such as your cheek book, adress book, and phone book. Each of databasses organizes data for easy storape and retrieval or access.

A simple axample of database is a business olient card file. Each client's name, address, and phone number an a separate card. Each record all data for person.

Before you can put data into a dotabase file, you must define its structure. This includes the names of fields in a record, the number of characters in a field, and the type of information (alphanümeric, nümeric, lopical so on) allowed in each field. In a database, all data for a farticulary entry is called a record, each record contains all data for one person. Each item of information within a record is called a field. The name or title of a field called field name, the type of entry allowed for each field called field type and the maximum number of characters in a field is called field with.

In a database manapemant system very impartant subjects are sorting, searching, indeksing, deleting. We shall study the mest populer indekxing methods in current use: The $B^{\pm}$ tree and Lineer hashing. Almost no other indexing schemes except $B^{\pm}$ trees and hashing are used today on large files, even for large files, the B^{+}trees is a two-disk-occess method. That ıs, $T_{f=s+r+b_{tt}+s+r+d_{tt}}$. Fetch time ($T_f$) assumes that the internal nodes of the B^{+}-trees are 2400-byte nodes so we use b_{tt}(block transfer time) and d_{tt} is data transfer time, s seek time to track andr, read time to sector of a disk.

This is much more officient than the pile file, where we had to read through half the file on average. It is more efficient than the sorted fie, where we had to use binary search. Since single-record fetches are so efficient and since range queries can also be answered with B^{+}-trees, this is the most popular indejing method in use today.

The reason B^{+}-trees are a two-disk-access method is that the fanout, fo, for internal nodes is high. If we have mem blocks in memory all but the parent-of-leaf level and the leaves fit in memory when

$$\text{mem} > \frac{bk}{(fo)^2}$$

With a fan-out of 140 and 10 megabytes, or 4167 blocks of 2400 bytes each, in memory, if bk is less than about 80 million buckets,

we have a two-disk-access method. None of teh files we use as examples exceed this limit. Some of our examples will be small enough for a one-disk-access method, but to make analyses ismpler, we shall always assume two disk accesses.

Since the records are stored in order within the leaves, primary B^+-trees can be used to list the records in order. This time is Tx and the number of data buket is bk.

$$T_X(\text{primary}) = bkx(s+r+dtt)$$

This is like reading the file randomly by bucket since the leaves ara not in order on the disk.

Hashing with Buckets and Chaining:

In summary, if the growth of a file can be predicted, hashing with buckets and chains is an excellent access method. It is fast for fetching, inserting, and deleting, but not for sequential operations. Hashing cannot be used for range queries.

Usually, rifht after a hash table has been created, fetching tekes about one disk access. This is the method of choice for many files and many database nanagement systems. When most of the operations on the file are individual-record fetches, hashing is the optimal file structure.

When there is a mixture of sequential operations, range searches, and individual-record fetches, a B^+-tree may perform better than a hash table. It 's a good idea to make a rough estimate of the time needed for a given mij of operations to decide whether to use hashing or B^+-trees or some combination.

Linear Hashing:

Several new methods have been proposed to avoid reorganization of hashed files. One such method is linear hashing (Litwin 1981). Linear hashing maintains a constant load fastor, even when many new records are added to the file. It does so by incrementally adding new buckets to the primary area. Thus as new records are added, new space for the records in, the primary area is also added. The load factor, which is the ratio between the number of records and the number of spaces for records in the primary area, remains constant. Maintaining a constant load factor as the file grows is a revolutionary idea.

Fetching a Record

In linear hashing, teh last binary bits in the hash number are used for placing the record. That is, we first calculate a large number from the key, just as before. Then we place the record in a bucket according to the last k binary digits in the hash number. For example, we could begin with eight buckets and use the last three bts in teh hash number that was calculated. If the number ended in 000, it would be placed in the first bucket. If it ended in 001, the second; in 010, the third; and so on.

When we exp and the table, we split an existing bucket which holds all the records which end in a particular k digits into two buckets using the k+1 last digits of the hash number. This way, for example, the bucket with records whose hash number ended in 010 would be split into one bucket with records whose hash number ends in 0010 and another bucket with records whose hash number ends in 1010 After such splits, some of thi buckets hold records where k digits are used and some where k+1 digits are used.

The splits are handled following an algorithm which we shall out line in treating insertion. For fetching, we need only know that some of the buckets use k digits, and some use k+1 digits, nd we can tell the difference by checking whether or not the last k digits are smaller than a given volue, the boundary value. The boundary value and the current value of k are kept in teh file header.

This method is very close in time to sorting and merging using one disk drive. With the sorting, we created sorted segments and then merged them, using at least two reads and writes of the whole file. With sorting, the large number of disk accesses comes during reading in for merging.

It helps to sort by hash value or else to partition by hash value, in order to construct a hsh table from ejisting large files. If we try to construct the table by successive insertions, it will take too much time.

In traditional hashing, reorganizing occurs often. So it is very i-portan to have an efficient way to create new hash tables

For linear hashing, we have to create the table only once. Still, we can save a considerable amount of time by sorting by the bits expected to be used in the table, instead of doing successive insertions. We gave an algırithm for deciding how many bits to use for creating a linear-hashing table.

BÖLÜM 1

GİRİŞ

Veri tabanı yönetim sistemleri, uzunca bir süre geniş verileri yöneten, güçlü computer programlarıyla, büyük computerlerin kullandığı sistemlerdi. Fiyat olarak da yüksek düzeylerdeydi. Bu günlerde, geliştirilen mikrocomputerler aracılığuyla etkin, düşük fiyatlı olarak az bir yatırımla gerçekleşebiliyor, uygulamaya sokulabiliyorlar. Böylece daha fazla veri tabanı yönetim sistemi kişisel ve iş dünyası için gelişebiliyor ve kullanılabiliyor.

Bir veri tabanı sistem programıdır. Bu tip programları yazmak için ençok kullanışlı dillerden birisi C programlama dilidir. Sistem programı yazılımı için C dilinin seçilmesinin sebeplerinden biri çok hızlı çalışmasıdır. C ile yazılan programlar Assemler dilinde yazılan programlar kadar hızlı çalışırlar ve daha kolay yazılabilirler. [2]

Geçmişte sistem programlarının çoğu Assembler ile yazılırlardı çünkü programlama dilleri yeteri kadar çabuk çalışan programlar yaratamıyorlardı.

C nin kullanımının diğer nedenleri bit, byte ve adresleri kolayca kullanabilmesi, giriş/çıkış fonksiyonlarını ve bellek yönetim fonksiyonlarını direk olarak kontrol edebilmesidir.

C dili ile yazılmış programlar çalıştırıldıklarında derlenirler, derlenen programlar daha hızlı çalışırlar. Örneğin BASIC'le yazılan bir programı çalıştırmak istediğiniz zaman, önce BASIC yorumlayıcısını ve daha sonra programınızı yükleye-

rek çalıştırmak zorundasınızdır.

Bir derleyici böyle değildir. Programınızı derlediği zaman onu obje kodununada çevirir. Böylece direk olarak Computer tarafından çalıştırılabilir. Programın çalışması için sadece ismini yazmanız yeterlidir.

Bu özelliklerinden dolayı, bir sistem programı olan VERİTAB Paket programı TURBO C (2.0) programlama dili ile yazılmıştır.

Takip eden bölümlerde, okuyucu için gerekli olacağı düşünülen teorik bilgiler verildikten sonra hazırlanan paket programın tanıtımına geçilecektir.

BÖLÜM 2

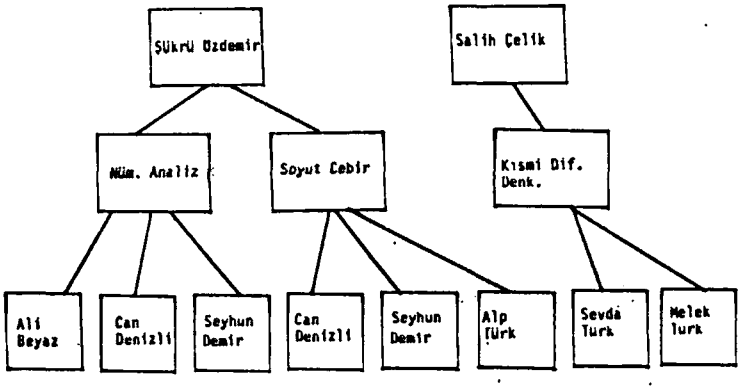
GENEL KAVRAMLAR

2.1. VERİ TABANI TANIMI VE TİPLERİ

Veri tabanı, belirli tarzda kullanılmış, faydalı bilgilerin bir kolleksiyonudur. Veri tabanı içinde depo edilen bilgilerin tipleri alfasayısal metinleri ve sayısal değerleri içerir. Verilerin düzenlenişine göre çeşitli veri tabanları tanımlayabiliriz. Veri tabanı yönetiminde en önemli iki tanesi hiyerarşik ve ilişkisel veri tabanlarıdır.

2.1.1. Hiyerarşik veri tabanı:

Bu veri tabanı, veri tabanını hiyerarşik ağaç modeli modeli üzerinde düzenler, veri tabanında veriler tanımlandığı gibi veriler arasındaki ilişkilerde tanımlanır. Hiyerarşik veri tabanının çeşitli modelleri mevcuttur. En basiti veri tabanındaki one-to-one (bire-bir) ilişkide gösterilen modeldir. One-to-many (bire-çok sayıda) many-to-many (çok sayıya-çok sayıda) olabilir. (Bire-bir) one-to-one ilişki şekil 2.1. de görülebilir.



Şekil 2.1. Hiyerarşik Veri Tabanında Birebir İlişki

Burada veri elemanları, ders veren elemanlar (öğretmenler), dersler ve öğrencilerdir.

Modeli aşağıdan yukarıya doğru tararsak öğrenciler ve dersler arasındaki bağlantının (ilişkinin) birebir (one-to-one) olduğu görülebilir. Çünkü ağaç yapısındaki her öğrenci sadece bir sınıfa aittir.

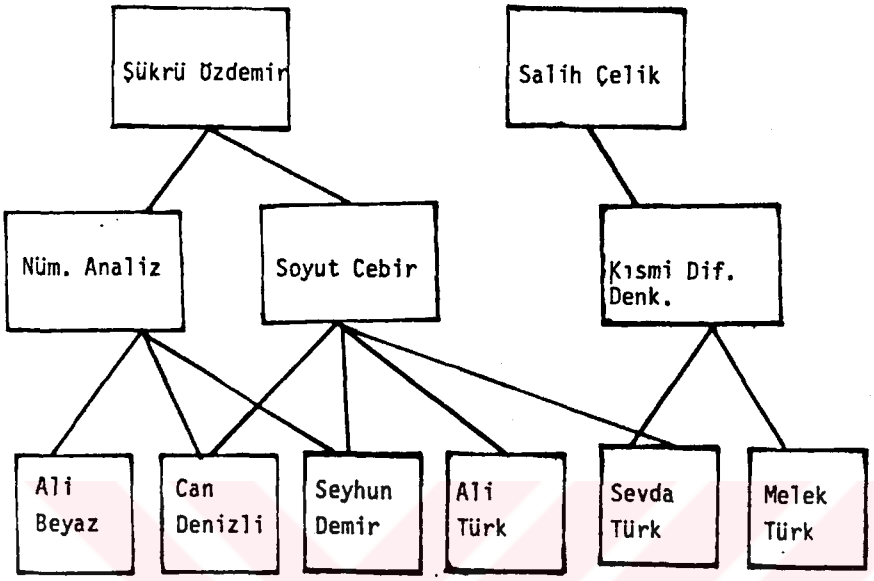
Burada öğrenci birden fazla ders aldığında, ağaç yapısında iki farklı veri elemanı olarak görülür (Can DENİZLİ). İki veri elemanı arasındaki bire-bir(one-to-one) ilişki, tek bir hatla gösterilir.

Aynı yöntem, sınıfla dersi veren öğretmen arasında da geçerlidir. Fakat ağaç yukarıdan aşağıya doğru taranırsa öğretmenlerden Şükrü Özdemir, iki sınıfa ders verirki bu da (one-to-many) bire-çok sayıda ilişkidir. Dersler ise Nümerik Analiz ve Soyut Cebirdir.

Başka bir gösterim de Network ağaç yapısıdır. (Şekil 2.2) Aynı ilişki, burada many-to-many (bire- çok sayıda) olarak tanımlanır.

Network ağaç yapısında gösterildiği gibi, Can Denizli adlı öğrenci iki ders alır. Her derste öğrenci sayısı birden fazladır.

Bu yapıda öğrenci birden fazla ders alsada ismi bir kez görülür. Yani her veri elemanı tektir, fakat bağlantılar birden fazladır. Network modelinde, veri elemanları az bile olsa diğerlerine göre daha karmaşıktır.



Şekil 2.2. Network Ağaç Yapısı

2.1.2 İlişkisel Veri Tabanı:

İlişkisel veri tabanının kullanımı, büyük bilgisayar sistemlerinde geliştirildi. Son zamanlarda, ilişkisel yapıdaki basitlikten ve mikrokompitürlerin gelişmesinden dolayı, yaygın uygulamalar kazandı.

Bu yapı satır ve sütun yapısından oluşan bir tablo şeklindedir. Her satır girdiğimiz bilgileri içerir. Örnek ilişkisel model Şekil 2.3. de gösterilmiştir.

Şükrü Özdemir	Küm. Analiz	Ali Beyaz
Şükrü Özdemir	Küm. Analiz	Can Denizli
Şükrü Özdemir	Küm. Analiz	Seyhun Demir
Şürkü Özdemir	Soyut Cebir	Can Denizli
Şükrü Özdemir	Soyut Cebir	Seyhun Demir
Şükrü Özdemir	Soyut Cebir	Mehmet Türk
Salih Çelik	Kıs.Dif.Denk.	Ali Türk
Salih Çelik	Kıs.Dif.Denk.	Sevda Türk

Şekil 2.3. İlişkisel veri Tabanı

Şekil 2.3. de görüldüğü gibi, sıfır veri tabanındaki elemanlar 9 satırlı ve 3 sütunlu bir tabloda görülebilir.

Her satır bir veri kaydı ve her sütun bir veri alanıdır. Her veri alanı, bir alfanümerik dizi veya etiketle gösterilebilir. (öğretmen, ders, öğrenci gibi.)

Veri kayıtları girilirken, her bir veri kaydına giriş sırasına göre bir numara verilir. Kaydın her bölümü farklı bir sütundadır. Sonuç olarak veri tabanındaki herhangi bir veri elemanı kayıt numarası ve alan ismiyle tanımlanabilir.

Telefon rehberi ilişkisel olarak Şekil 2.4. de gösterilmiştir.

1	Ahmet Bora	296	Beşiktaş	160 02 74
2	Ali Beyaz	212	Aksaray	523 01 13
3	Can Turkgenç	503	Taksim	176 18 20
4	Şevim Gündüz	315	Fatih	167 12 10
5	Reyhan Gökçe	415	Haseki	170 15 19
6	Canan Tanlı	618	Çapa	516 80 88

Şekil 2.4. İlişkisel Veri Tabanı Telefon Rehberi

2.3. İLİŞKİSEL VERİ TABANI DOSYASININ TEMEL BİLEŞENLERİ

İlişkisel veri tabanı iki temel bileşenden oluşur.
Birincisi veri kayıtlarının yapısını tanımlar.
İkincisi ise gerçek yapısını kapsar.

2.3.1. Veri Kayıtlarının Yapısı:

1. Veri Kaydı:

Veri kaydı, giriş için veri bölümlerini alır. Telefon rehberi örneğinde, (Şekil 2.4.) tüm bir isim ve telefon numarası, bir kaydı oluşturur.

Data (Veri) kayıtları genellikle giriş sırasına göre düzenlenir. Her kayıt veri tabanına eklendiğinde sıralı bir

kayıt numarası ile gösterilir. Daha sonra kullanıcı veri kayıtlarını kayıt numaraları ile teşhis edilebilir.

2. Veri Alanı:

Veri alanı bir veri kaydındaki tek bir veri bölümünü tutmak için, depolama ünitesidir. Her veri alanına bir isim verilir. Alan isimleri harfler veya kesin sembollerdir. Veri alanının içeriği alfanümerik dizi veya sayısal değer alabilir, sayı değeri tamsayı veya decimal haneli sayı olabilir.

Alfanümerik dizinin uzunluğu ve değer için ayrılmış yerlerin sayısı, veri alanı kullanılmadan önce, veri yapısında açıkca tanımlanmalıdır.

2.3.2. Veri Tabanının Yapısı:

Veri tabanı yapısı, bir veri kaydındaki her alanının detaylı tanımını kapsar.

Alan ismi : veri alanının ismi

Alan tipi : veri alanının tipi (çeşidi)

Alan genişliği : veri alanı boyutu.

Bu tanımlanan yapı, çeşitli amaçlara hizmet eder. Veri işletimi esnasında, özel olarak verilmiş olan ismi göndermek ve çağırmak için kullanılabilir.

Veri alanına maximum sayıya göre uzunluk verilmelidir. Örnek olarak en uzun ismin uzunluğunun değeri ne işe veri alanına, buna göre uzunluk verilmelidir.

BÖLÜM 3

İLİŞKİSEL VERİ TABANININ UYGULAMALARI

İlişkisel veri tabanınıda şu işlemleri yapacak fonksiyonları oluşturabiliriz.

- Veri tabanı içeriklerinin tutulması ve güncelleşmesi.
- Bir grup halinde özellikleri verilmiş ve bu şekilde biraraya toplanmış olan veriyi yerleştirmek ve bulup getirmek.
- Verileri önceden verilen bir sırada düzenleme ve tasnif etme.
- Veri yapılarında dolaylı indekslenmeye sahip olmak için farklı veri yapılarını bağlamak.

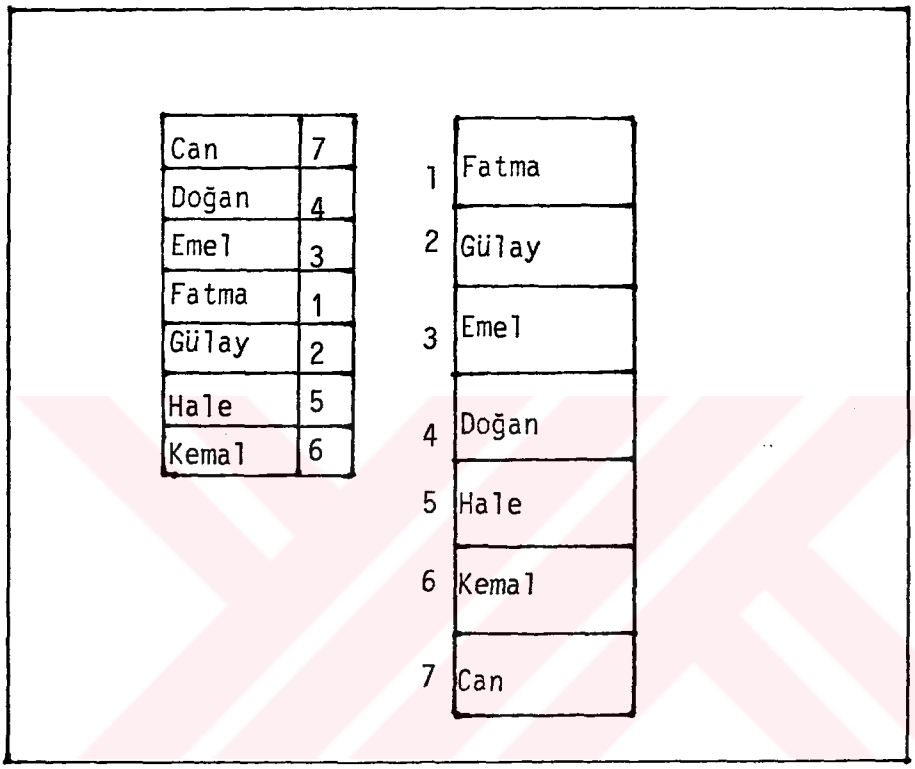
Burada veri tutulması demek, veri tabınına veri eklenmesi, veri tabınının içeriğinin kısmen veya tam olarak değiştirilmesi veya veri maddelerinin silinmesi demektir.

Bir veri tabanında dosyaya yapılan kayıt eklemeleri, dosyanın sonuna yapılır.

Diğer önemli fonksiyonda kayıt nosu ve alan ismiyle verinin yerleştirilmesi veya tekrar bulunup getirilmesidir.

İlişkisel veri tabanında tasnif etme ve yeniden düzenlemede, önemli bir özellik daha, farklı dosyalardaki veri elemanlarının birbirleriyle olan ilişkileri ve aralarındaki bağlantılardır.

Yukarıda anlatılanlarla bağlantılı olarak bir indeks dosyadaki bilgilerden, ana dosyaya, random (doğrudan) erişim olayı Şekil 3.1. de gösterilmiştir.



Şekil 3.1. İndeks ve Ana Dosyanın Birlikte Kullanılışı

3.1. Verilerin Organizasyonu:

Verilerin yapısına bağlı olarak, veri dosyalarının birçok farklı tipi vardır. Her biri verinin özel bir çeşidinin depolanması için kullanılır.

Bir veri tabanı çeşitli veri tiplerinin birkaç veri dos-

yasını kapsar. Çünkü veri dosyasındaki bilgi genellikle dışsal bir bellek alanında toplanır. Veri dosyası sık sık disk dosyası olarak da adlandırılır.

3.2. Veri Dosyaları (Disk dosyaları):

Bir disk dosyası bellekte bilgiyi saklamak için ayrılmış bir bellek alanı temsil eder.

Disk dosyalarına birkaç örnek vermek istersek şöyle sıralayabiliriz.

- Veri tabanı dosyaları
- İndeks dosyaları
- Veri tabanı memo dosyaları
- Komut dosyaları
- Format dosyaları

3.2.1. Veri Tabanı Dosyaları:

Veri tabanı dosyası ilişkisel veri tablolarına karşılık gelir. Dosyanın içeriğinde veri tabanında kullanılacak veri alanları vardır. Bunlar Alfanümerik karakter dizisi, nümerik alanlar ya da tarihsel alanlar olabilirler.

3.2.2. Veri Tabanı Memo Dosyaları:

Veri tabanı dosyasına benzer. Memo olarak adlandırılan geniş metin bloklarını kapsar. Veri tabanı memo dosyaları, veri tabanı için ilave bellek sağlar.

Memo metni, veri tabanı dosyasındaki veri alanı olarakda tanımlanabilir. Fakat veri tabanı içeriği kendisinden ayrı olan memo alanında depolanır.

3.2.3. İndeks Dosyaları:

Veri tabanı dosyasını, özgün bir yöntemde indeksler ve tasnifleme işleminin etkisini meydana çıkarır. İndeks dosyası, indeks işlemi için gerekli yeri sağlar. Veri seti uygun bir sarada kullanılabilir. Tercih edilen şey bu sıranın kayıtların giriş sırası olmasıdır.

3.2.4. Komut Dosyası:

Komut dosyası işleme sokulan komutların, toplamını saklar. Komutlar, komut dosyasından girilebilir veya komut dosyası döküman yapıda olmayan kelime işlem programıyla yaratılabilir.

3.2.5. Format Dosyaları:

Çıkış formatını (yazılım) tayin eden bilgiyi depolar.

BÖLÜM 4

Bu bölümde iki önemli veri düzenleme operasyonu tanıtılacaktır. Sıralama ve indeksleme. Burada üzerinde durulacak konular başlıca şunlardır.

4.1. Bir veri tabanı kütüğünde veri organizasyonu.
4.2. Veri kayıtlarının sıralanması
4.3. Veri tabanı kütüğünü indeksleme

4.1. Bir Veri Tabanı Kütüğünde Veri Organizasyonu:

Bir veri tabanı kütüğü veri kayıtları ve veri alanlarından oluşur. Bir veri kayıtındaki veri alanlarının tanımları, kütük yaratıldığında belirlenir. Kayıtlar girildiği düzende saklanır ve herbir kayıta daha sonra başvurulmak için bir kayıt numarası ayrılır. Kayıtların gelişigüzel olarak düzenlenmesi bizim ihtiyacımıza uygun olmayabilir. Yani alfabetik olarak düzenli değildir. O halde kayıtlar istenilen amaca göre sık sık yeniden düzenlemeye tabi tutulurlar.

Sıralama işleminde, belirlenen anahtar alan tarafından alfabetik, nümerik veya tarihe bağlı olarak, kayıtları sıralamak mümkündür.

Alfabetik alanların sıralanmasının artan düzende olması isteniyorsa, sıralama A dan Z ye olur. Anahtar alan olarak tarih alanı seçilmişse, ilk önce gelen tarihli kayıttan önce yerleşir.

Nümerik değerler artan düzende sıralanmışsa en küçük değerli kayıtlar büyük alanlardan önce yerleşecektir.

Sıralama bir tek alan üzerinde yapılabileceği gibi ikinci alan içinde uygulanabilir. Bu durumda anahtar alanda aynı olan veriler için ikinci alandaki veriler de kendi aralarında artan yada azalan bir sıraya konulurlar.

Sıralama işlemi yapılırken veri alanlarının sayısı büyükse (genişse) ve veri sayısıda fazlaysa bu operasyon çok uzun işlemler gerektirebilir.

İstenilen herhangi bir alana göre alfabetik sıraya konulabilirler ya da kayıt numaralarına göre indeks yaratılarak dosyadaki verilere (bilgilere) daha kolay erişme olanağı sağlanmış olur.

4.2. Veri Kayıtlarının Sıralanması:

Veri kayıtlarını artan veya azalan düzende yeniden sıralamak için bir sıralama operasyonu kullanılabilir. Sıralama işlemine başlamadan önce çalışma kütüğü olarak kullanılacak ayrı bir dosya yaratılır. Bunu yapmakta amaç sıralama işlemi esnasında, sıralamaya tabi tutulacak kayıtların bu dosyaya yazılmasıdır. Böylece orjinal dosyamız olduğu gibi bozulmadan kalmış olur. Sıralama (sort) işlemi bu dosya üzerinde yapılır. Sonra kayıtlar belirlenmiş anahtar alanlara göre yeniden düzenlenirler. Sonuçlar bu dosya üzerindedir. Kayıtlara erişirken, bu dosya ile ana dosya (orjinal dosya) arasında ilişki kuralarak erişim işlemi sağlanır. Böylece kayıtlara direkt olarak erişilmiş olur.

4.3. Bir Veri Tabanı Kütüğünü İndeksleme:

Veri kayıtlarını düzenlemek için kullanılan diğer bir metotta indekslemedir.

İndeksleme operasyonu bir target dosya (kopya edilen) yaratır ki bu dosyadaki kayıtlar belirlenmiş anahtar alana göre nümerik, tarihsel veya alfabetik olarak artan düzende tanzim edilirler. Bu target dosya indeks dosyası olur. Bu dosya belirleyici tip olarak indeks olarak diğer dosya tiplerinden ayrılır. İndeks dosya sadece anahtar alanları ihtiva eder ve kayıt sayılarına karşılık gelir. İndeks dosya aktif veri tabanı kütüğündeki kayıtları yeniden düzenlemek için kullanılır.

İndeksleme işlemi sıralama işlemine benzer fakat indeksleme işlemi, saralama işlemlerinden daha hızlıdır.

BÖLÜM 5

DOSYALAMA İŞLEMLERİ

- Yeni bir dosya yaratılması
- Dosyaların listelenmesi
- Dosyaların silinmesi

5.1. Yeni Dosya Yaratılması:

Veri tabanını oluşturacak bir ana dosyanın yaratılması bu kısımda gerçekleştirilir. İlk olarak dosya yaratılır ve daha sonra bilgiler buraya kaydedilir.

Kayıt numarası bilinmiyorsa isim arayarakta kayıtın kendisine erişmek mümkündür.

5.2. Dosyadaki Bir Kaydın Harflerle Aranması:

Dosyada bulunan kayıtlara isimlerinin baş harfleriyle erişilir.

KA yazdığımız zaman dosyada ismi Ka ile başlayan kayıtlar bulunup getirilir.

5.3. Dosyaların Listelenmesi:

DOS daki Dır komutunun karşılığı olarak o anda çalışılan ddırectorydeki dosyaların listesini verir.

5.4. Dosyaların Silinmesi:

Programda kullanılan indeks dosyaları ve ana dosyalar silinir.

BÖLÜM 6

İNDEKSLEME METODLARI

Bugün veri tabanı yönetim sistemlerinde en çok kullanılan indeks metodları $B^{\pm}$ ağaçları ve hash adresleme metodudur.

6.1. $B^{\pm}$ Ağaçları

Bugün en çok kullanılan indeksleme metodlarından biri $B^{\pm}$ ağaçlarıdır. $B^{\pm}$ ağaçlarının birkaç versionu vardır. Bunlar $B^{\pm}$ ağaçları, B^{*} ağaçları, ve B ağaçları olarak adlandırılırlar. B^{+} ağaçları bu yapının standartlaşmış ismi haline gelmişlerdir.

Her hangi bir ikili ağaç-özellikle ikili arama ağacı bir B^{+} ağacı değildir. İkili ağaçlar bir düğümde en fazla iki çocuk bulundururlar. B^{+} ağaçlarında ise 100 den fazla çocuk vardır. Bütün yapraklar ise aynı seviyededirler. İkili arama ağacı farklı seviyelerde birçok yaprağa sahiptir.

Diyelim ki müşteriler içinde borcu 1000 dolardan fazla fakat 100.000 dolardan az olanlarını bulmamız isteniyorsa. Burada cevap değerlerinin sırasında yatmaktadır.

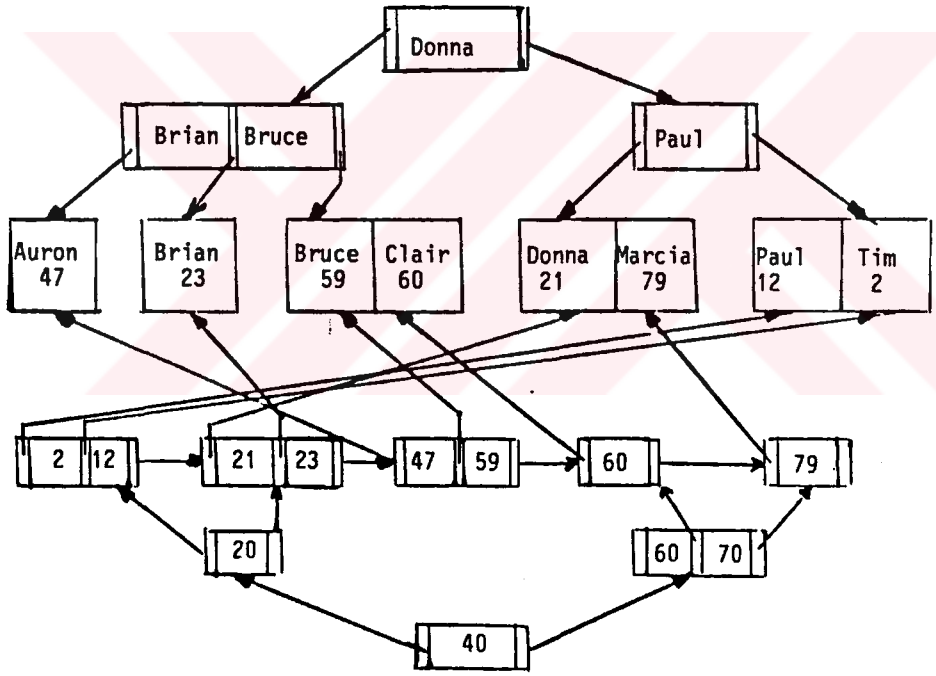
6.1.1. V Basamağında (seviyesinde) bir B^{+} ağacının Tanımı:

1) Kök bir yaprak olmadıkça en azından iki çocuğa sahiptir.

2) Kök hariç bir iç düğüm 2v den daha fazla anahtar alana sahip olamaz. İçdeki doğümler daha aşağı seviyelerdeki düğümlerin anahtar alanlarını ve adreslerini içerirler.

3) Bütün yapraklar aynı seviyededirler. Eğer B^+ ağaçları birincil indeks oluşturmak için kullanılıyorsa ağacın yaprakları, veri kayıtlarını ihtiva eder. İkincil indeks olarak kullanılıyorlarsa yapraklar anahtar alanları ve disk adreslerini ihtiva ederler. Ayrıca yapraklar sıralı aramalar için, bir sonraki yaprağın adresini de içerirler.

4) k tane key'i olan bir iç düğüm k+1 tane çocuğa sahiptir.



Şekil 6.1. Aynı dosya için birincil ve ikincil B^+ ağaç indeksleri. Birincil indeksin yaprakları dosya kayıtlarıdır. İkincil indeksin yaprakları sadece anahtar alanları ve dosyadaki kayıtlara karşılık gelen disk adresleri içerirler.

Bir yaprak düğümde bulunan kayıt sayısı buket faktörü olarak adlandırılır. Yaprak düğümleri bir kaç blok ihtiva eden dosya buketleridir. Kayıtlar, anahtar ve adreslerden daha çok yer kaplar. Bu durum B^+ ağaçlarının neden tercih edildiğini daha iyi açıklar. B^+ ağacındaki bir iç düğümün ortalama çocuk sayısı, fon.out olarak adlandırılır.

Bir primer indeks, kayıtların depo edilmesini sağlar. Eğer bir birincil indeks aynı zamanda kümeleme indeksi ise, kayıtlar anahtar alanın değerine göre buketler halinde depo edilirler. Belli bir buketteki yığınlar birbirine yakın anahtar alan değerlerine sahiptirler. Böylece indeks belli bir yığındaki en düşük ve en yüksek anahtar alanı göstermeye yarar. Bu nedenle kümeleme indeksi bir sparse (seyrek) dizi olarakta adlandırılır. Yapraklarında verileri olan bir B^+ ağacı bir sparse dizidir.

Çoğunlukla bir dosya için birçok ikincil indeks kullanılabilir. Örneğin ikinci isme göre bir birincil indeks kullanıldığında kayıtları soyadına göre oluşturabiliriz. Aynı zamanda sicil numarasına göre ikincil indeksimiz olabilir. Kayıtlar ikinci isme göre oluşturulduğu halde sicil numarasına göre arama yapılabilir. İkincil indeks doğum tarihine göre de kurulabilir ve biz herhangi bir yıldan önce doğan bütün kayıtları inceleyebiliriz.

Dosyadaki her alan için ikincil indeks oluşturursak ağacın yapraklarında dosyadaki bütün bilgiler tekrar edilir. Bu dosyanın kendisinden daha çok yer kaplayacaktır. Zira Ağaç indeksleri göstergeler ve ağacın daha üst seviyeleri için yere (belleğe) ihtiyaç duyarlar. Bu kayıtlara erişimi kolaylaştırır, fakat hafıza kullanımı açısından bedeli yüksektir. Ayrıca yeni bir giriş olayında bütün indeks yeniden düzenlenmelidir.

Dosya tasarımında önemli bir karar dosyadaki indeks için en uygun olanı seçmektir.

Veri tabanı yönetimlerinde hash yada zincirleme metodları birincil indeks olarak kullanılabilirken, B^+ ağaçları ikincil indeks olarak kullanılır.

6.1.2. B^+ ağaçları İçin Arama Algoritması:

1) Kökden başla

2) Her bir düğümde bir sonraki seviye için k+1 tane pointer (gösterge) vardır. ek olarak k tane de anahtar alan vardır.

anahtar alanlar $c_1, c_2 \ldots\ldots c_k$

göstergeler $p_0, p_1 \ldots\ldots p_k$

Aradığımız kaydın anahtar alanının değerinin x olduğunu kabul edelim. a) $c_i < x \leq c_{i+1}$ $(1 < i \leq k$ için) ise p_i yi bir sonraki seviyeye kadar takip et.

b) Eğer $x < c_1$ ise p_0, ı, ağacın bir sonraki seviyesine kadar takip et.

c) Eğer $c_k \leq x$ ise p_k yı bir sonraki seviyeye kadar takip et.

3) Yaprak seviyesine erişilene kadar 2. adımı tekrarla.

Dosya açıldığı zaman yaprağın ebeveyni seviyesinin ve veri düğümlerinin hafızaya alındığını düşünerek B^+ ağaçları kullanıldığında bir kayda erişme zamanı yaklaşık

$T_f = s + r + btt + s + r + dtt$ dir.

Burada B^+-ağacının iç düğümlerinin 2400 byte düğüm oluşturduğu kabul edildi. Burada b_{tt} block'a transfer süresidir. Yaprakların pekçok bloktan oluşan buketler olduğu farzedilerek d_{tt} kullanıldı. Burada d_{tt} veriyi buketlere transfer etme süresi, s ve r ler ise kayıtlar bir disk den arandığında, sırasıyla track'e erişme ve sector'e erişme zamanıdır. Eğer yapraklar sadece 2400 -byte block ise formül

$$T_f = 2x\ (r+s+\ btt)$$

olarak basitleşir.

6.2. BUKETLERLE VE ZİNCİRLERLE HASH ADRESLEME

Hash adresleme tekniğinin tanımını vermeden önce basit bir örnekle algoritmayı açıklayalım.

1000 tane tamsayımız olduğunu ve bu tamsayıların dosyadaki kayıtların anahtar alanları olduğunu kabul edelim. Anahtar alanı 1 olan kaydı dosyanın 1. kaydı, anahtar alanı 2 olan kaydı dosyanın 2. kaydı, anahtar alanı 3 olan kaydı, dosyanın 3. kaydı olacak şekilde, kayıtları diske yazmaya devam edelim. Bu durumda anahtar alanın değeri, aynı anahtar alanlı kaydın disk adresini verecektir.

Bu metod One-disc-access metodudur ve B^+ ağaçlarında kayıtların tek tek aranması işleminden daha hızlıdır. Hash adresleme metodu, bu metodun genelliştirilmesidir.

Hash adresleme algoritması, anahtar alanlarının değerlerine göre bir hash tablosu oluşturmak ve hash tablosunda bulunan anahtar alanların değerlerinden disk adreslerini hesaplamaktır. Bazen direk erişim, rasgele erişim yada anahtar alandan adrese tranformasyon (key to adress transformation) olarak adlandırılır.

6.2.1. Buketler:

Bellekte kayıtları depolamak için ayrılan alanlara birincil alan denir. Buketler birincil alanın parçalarıdır. Bir buket birkaç bloktan oluşabilir. Buketdeki blok sayısına buket faktörü (Bkfr) denir.

Birincil alandaki buket sayısı M olsun. Anahtar alanının alfanümerik olduğunu farzederek, her bir karakterin ASCII kodu toplanarak büyük bir sayı elde edilir. Bu sayı M'e bölünerek kalanı alınır. Elde edilen sayı 0 ile M-1 arasındadır. Bu sayı birincil alanda bulunan buketlerden birine karşılık gelecektir. Hesaplanan değeri x olan bir anahtar alan, x numaralı buket içine yerleştirilir. Birden fazla anahtar alan için hesaplanan değer aynı olabilir. Buket de yer olduğu sürece bir problem oluşturmaz.

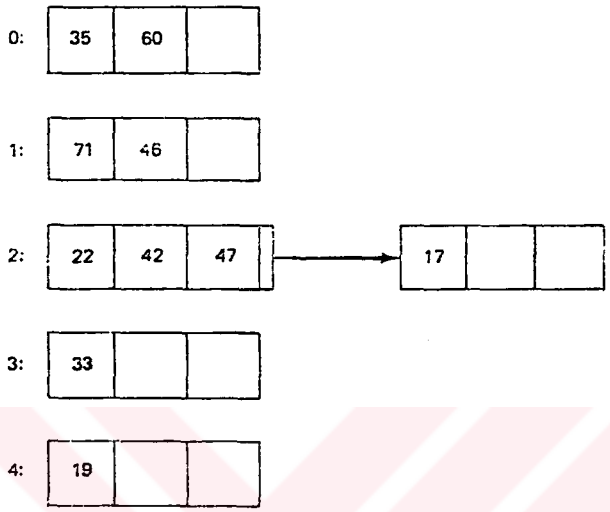
Veriler düzenli bir şekilde dağılmadıkları için, hash fonksiyonları hiçbir zaman sıra oluşturmak için elverişli değildir.

6.2.2. Ayrı Zincirleme:

Buketler dolduğu ve anahtar alanın değeri dolu buketteki kayıtların anahtar alanı ile aynı hash değerine sahip olan bir kayıt, bukete yerleştirmek istendiği zaman yeni buketin adresini, özel bir alana ekleriz. Bu alana overflow alanı denir. Overflow alanı birincil alandan bağımsızdır.

Birincil alandaki buketler dolduğu zaman birinci alana overfow buketinin adresinin eklenmesine zincirleme denir.

Aşağıdaki şekilde buketler ve zincirleme hash adresleme tekniği gösterilmiştir. Primer alanımızda 5 buket var. Hash fonksiyonu (h(x)) mod 5 dir. Buket faktörümüz ise üçtür.



Şekil 6.2.

Buketler ve ayrı zincirlerle hash metodu. Burada buket faktörü 3 ve hash fonksiyonu h(x)=xmod 5 dir.

6.2.3. Yükleme Faktörü (Load faktör):

Hash fonksiyonları kayıtları rasgele dağıtmış olmalarına rağmen aynı hash değerine sahip birden fazla kayıt olabilir. Birincil alanda yalnızca dosya için gereken bellek kadar yer ayrılırsa zincir sayısı artacaktır. Overflowu sınırlandırmak amacıyla birincil alanda dosyanın depolanması için gerektiğinden daha fazla yer ayrılır. Bu nedenle birincil alanın bir kısmı dolu olacaktır. (% 70, % 90 gibi).

Yükleme faktörü. Dosyadaki kayıt sayısının, birincil alanda kayıtlar için ayrılan alan sayısına (MxBkfr) bölünmesiyle elde edilir.

Lf = n/MxBkfr

6.2.4. Buketleri ve Zincirleri Kullanarak Kayıtlara Erişme:

Değerlerin dağılımı düzenli ve kayıtlar için bellekte ayrılan alan yeteri kadar büyükse, dosyadaki kayıtlara tek tek erişim açısından hash metodu çok etkindir. Erişim zamanı T_F= s+r+dtt dir. (s,r tanımları 6.1.2 de verildi.) dtt; bir buket, birden fazla bloktan oluştuğunda veriyi buket içine transfer etme süresidir.

Genel olarak küçük yükleme faktörü daha az overflow alanı demektir. Buna rağmen kullanılmayan alan sayısı fazladır. Yükleme faktörü % 50 olarak seçildiğinde erişim zamanı mükemmeldir, fakat diskin yarısı boştur. Büyük yükleme faktörü daha büyük dtt demektir. Her bir bukete erişme süresi artmış olur.

Herhangi bir kaydı aramak için önce birincil alandaki kayıtla ilgili bukete gidilir. Kayıt burada değilse, ilk overflow buketine bakılır. Kayıt bulunulana kadar bu işleme devam edilir. Eğer kayıt buradaysa ortalama olarak zincirdeki buketlerin yarısına bakıldığında bulunur. Buna başarılı arama. (successful search)

Dosyada olmayan bir kaydı aramak için gereken süre kaydın dosyada olmadığını anlamak için bütün zincirin taranmasından ötürüdür. Buna başarısız arama (unsuccessful search) denir. Overflow olduğunu ve zincirin uzunluğununda x olduğu kabul edilirse erişme zamanı

T(successf l)= s+r+dtt+(x/2)x(s+r+dtt)

T_F(uns ccessful)= s+r+dtt+xx(s+r+dtt)

Verilen buket faktörleri ve yükleme faktörleri ile erişim sayıları Şekil (6.3.) de gösterilmiştir.

	10%	20%	30%	40%	50%	60%	70%
1	1.0048	1.0187	1.0408	1.0703	1.1065	1.1488	1.197
2	1.0012	1.0088	1.0269	1.0571	1.1036	1.1638	1.238
3	1.0003	1.0038	1.0162	1.0433	1.0898	1.1588	1.252
4	1.0001	1.0016	1.0095	1.0314	1.0751	1.1476	1.253
5	1.0000	1.0007	1.0056	1.0225	1.0619	1.1346	1.249
10	1.0000	1.0000	1.0004	1.0041	1.0222	1.0773	1.201
20	1.0000	1.0000	1.0000	1.0001	1.0028	1.0234	1.113
50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0007	1.018

	10%	20%	30%	40%	50%	60%	70%
1	1.0500	1.1000	1.1500	1.2000	1.2500	1.3000	1.350
2	1.0063	1.0242	1.0520	1.0883	1.1321	1.1823	1.238
3	1.0010	1.0071	1.0216	1.0458	1.0806	1.1259	1.181
4	1.0002	1.0023	1.0097	1.0257	1.0527	1.0922	1.145
5	1.0000	1.0008	1.0046	1.0151	1.0358	1.0699	1.119
10	1.0000	1.0000	1.0002	1.0015	1.0070	1.0226	1.056
20	1.0000	1.0000	1.0000	1.0000	1.0005	1.0038	1.018
50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.001

Şekil 6.3. Buket Faktörleri ve Bir Kayda Erişim Sayısı

6.2.5. Sıralı İşlemler

Hash fonksiyonlarının kayıtları rasgele dağıtmış olmalarından ötürü, hash metodu bir aralık aramada yada bir dosyanın sıralı düzende listelenmesi için kullanışlı değildir. [1]

Veri kayıtlarının hash metoduyla oluşturulduğunu ve anahtar alanında sicil numaraları olduğunu kabul edelim. Bu kayıtlar içinde sicil numarası 000-000000 ile 000-99-9999 arasında olanları bulmak istiyorsak, 0 ile 999.999 arasındaki 1 milyon ayrı rakam için, 1 milyon arama yapmak zorundayız. Bunlardan çoğu da dosya içindeki kayıtlara karşılık gelmeyecektir.

Kayıtlara tek tek ulaşım açısından hash metodu etkin olmasına rağmen sıralama işlemlerinde, blok halinde aramalarda B^+ ağaçları daha kullanışlıdır.

6.3. LİNEER HASH ADRESLEME METODU

Hash adresleme metoduyla düzenlenmiş dosyaların tekrar organize edilmesinden kaçınmak için geliştirimiş bir çok yeni metoddan biridir (LİTVİN 1981). Bu metodla, dosyaya yeni kayıtlar eklerken, primer alana yeni buketler ekleyerek Load faktörü sabit tutulur.

Kayıtların yerleştirilmesi: Daha önceki bölümde anlatıldığı gibi, önce anahtar alanın ASCII kodlarından büyük bir sayı hesaplanır. Hash sayısının binary dijitlerinin son k dijitine bakarak buketlere yerleştirilir. Buketlere yerleştirilirken bir sınır değerle mukayese edilerek, bazıları son k dijite, bazıları ise son k+1 dijite göre buketlere gönderilirler. Sınır deger ve dijit sayısı kayıtlar buketlere yerleştirilmeden önce belirlenmiştir.

Şimdi sınır değer ve dijit sayısının nasıl tespit edildiğini anlatacağız.

Önce Primer alandaki buket sayısı (M=n/Bkfrxlf)) bulunur.

$$2^k \leq M < 2^{k+1}$$

şartını sağlayan k, dijit sayımızdır.

$M-2^k$ ise sınır değerdir.

Hesaplanan hash sayıları içinde son k dijit'e bakılır $M-2^k$ nın değerinden küçük olanlar, son k+1 dijit değerine göre, $M.2^k$ dan büyük yada eşit olanlar ise k dijit değerine göre buketlere yerleştirilirler.

6.4. GİRİŞ (INSERT)

Lineer hash metoduyla giriş algoritmasını vermeden önce, Buket faktörü 10 ve load faktörü % 70 ile bir tablo oluşturalım. İlk önce kayıt sayısını 56 olarak kabul edilim. Primer alandaki buket sayısı 8 dir. Dosyaya 7 kayıt daha ilave edildiğinde, primer alana 1 buket daha eklenir. Bunu genelleştirmek istersek, dosyaya (LfxBkfr) kadar kayıt eklendiğinde, primer alana bir buket daha eklenir. Kayıtlar buketlere yerleştirilirken bilinmesi gereken şey sadece kaç binary dijitin kullanılacağıdır. Primer alanı buketlerle büyütme sırasında ise, eski buket modife edilir ve yeni buket eklenir.

Lineer Hash metodu ile giriş (insert) algoritması:

1. Yeni kaydı doğru bukete yerleştirmek için yerini tespit et.
2. Buket dolu ise yeni bir buketi primer alana ekle. Kaydı yeni bukete koy. Bu yeni buketin adrasini zincirdeki son bukete ekle.
3. Eğer verilen Load faktörü için gerekli kayıttan daha fazla kayıt varsa, primer alana yeni bir buket ekle ve sınır değeri 1 arttır.

Yukarıdaki algoritmayı daha iyi anlaşılması için bir örnekle açıklayalım. Şekil (6.4.)

Buket faktörünün 3 ve Load faktörünün % 67 (2/3) olduğunu kabul ederek 4 buketle algoritmayı anlatmaya başlayalım. 8 kayıt Load faktörünü tam olarak gerçekler 8 kayıt eklendikten sonra Aşağıdaki adımları tekrar ederiz.

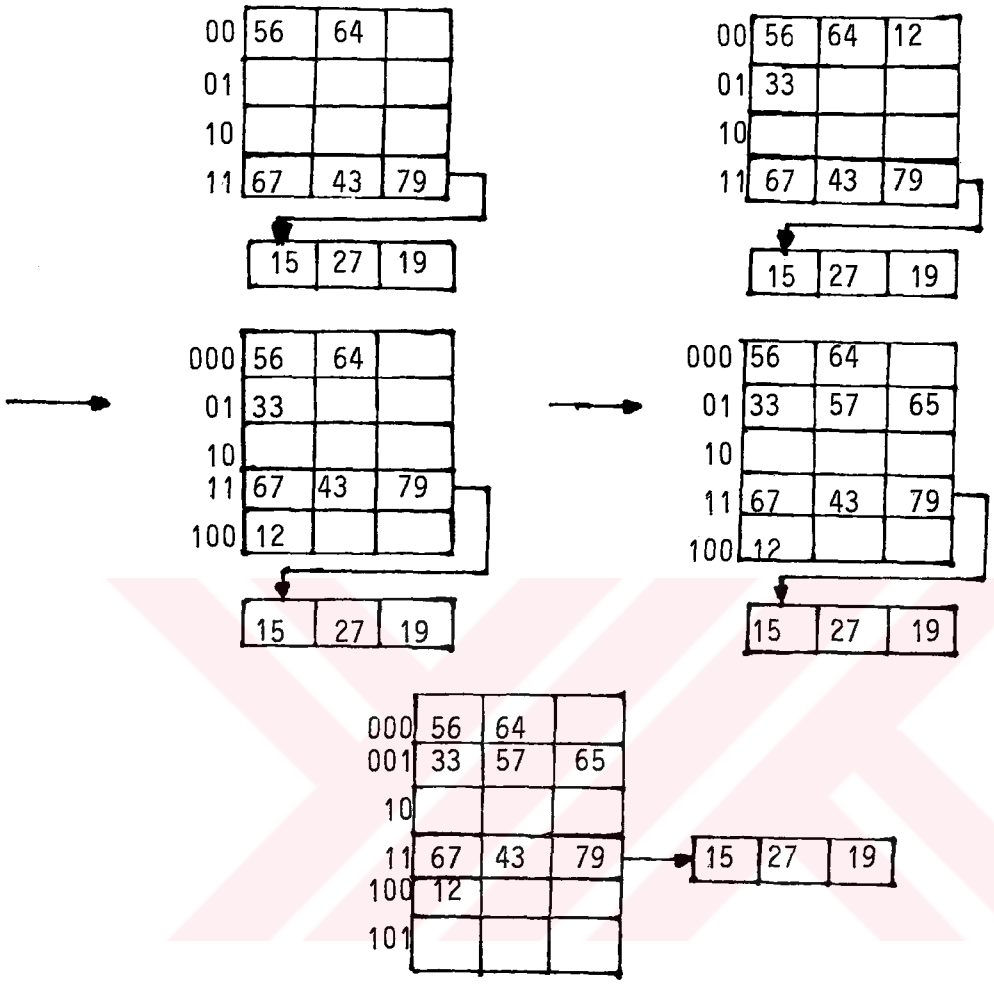
1. İki kayıt ekle
2. Birincil alana bir buket daha ekle

Şekil 6.4. de de görülebileceği gibi

56, 67, 43, 79, 15, 27, 19, 64, 12, 33, 57 ve 65 sayıları kullanıldı.

56 = 011 1000
67 = 100 0011
43 = 010 1011
79 = 100 1111
15 = 000 1111
27 = 001 1011
19 = 001 0011
64 = 100 0000
12 = 000 1100
33 = 010 0001
57 = 011 1001
65 = 100 0001

Bu sayılar overflow ve tablo genişlemesinin bağımsız olaylar olduğunu açıklamak için özellikle seçilmişlerdir. Overflowlar tablonun büyümesi üzerinde bir etki yapmazlar. Tablo büyümesi ise Overflow olayını azaltmaz. Şekil 6.4.



Şekil 6.4. Lineer hash metodu ile basit bir örnek. 8 kayıt ve 4 birincil alan buketi ile başlanır. Yükleme faktörü 2/3 ve buket faktörü 3 tür.

6.5. ARAMA

Lineer hash metodunda kayıtlara erişmek için bilinmesi gereken tek şey, keylerin bazılarınn k, bazıların ise k 1 dijitinin, kullanılarak buketlere erişmektir. Lineer hash metoduyla ka bir kayda erişme süresi yapılan deneylerle birçok kez mukayese edildi. Örneğin: Buket faktörü 50 load faktörü % 75 olarak alındığında, dosyada olan bir kayda 1.05 erişimde ulaşılmıştır. Bu başarılı arama olarak adlandırılır. Dosyada olmayan bir kaydı arama ise başarısız aramadır ki, dosyada bir kaydın olmadığı 1.27 erişimde bulunmuştur.

T_F (başarılı) = 1.05x(s+r+dtt)

T_F (başarısız) =1.27x(s+r+dtt)

Kayıtlara erişme süresi, lineer hash adresleme metodunda, dosyadaki kayıt sayısına bağlı değildir. Bu lineer hash metodunun, diğer genel hash metodlarına karşın bir avantajıdır. Diğer hash metodlarında dosyadaki kayıt sayısı arttırkça sırasıyla load faktörü ve erişim süreside fazlalaşır.

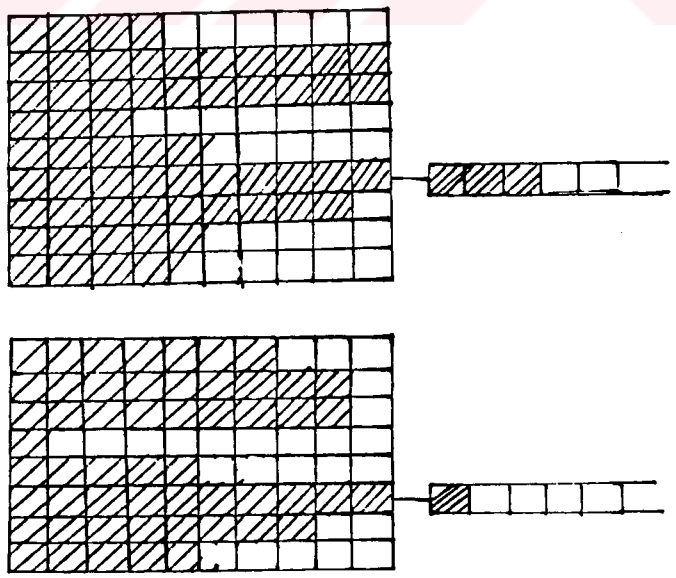
Lineer hash metodu için Arama Algoritması

1. Hash fonksiyonunu hesapla
2. Hash fonksiyonunun son k dijitine bak eğer son k dijit sınır değerden küçükse, kaydı hash fonksiyonunun son k+1 dijitine göre bukete yerleştir. Eğer son k dijit sınır değerden büyük yada eşitse kaydı buketteki k'ıncı bölgeye yerleştir.
3. Genel hash tekniğindeki gibi overflow zincirlerini takip et.

6.6. SİLME

Lineer hash metodunda silme, ekleme olayının tersidir. Ekleme olayında olduğu gibi önceden dijit sayısı hesaplanır (k). Silinecek olan anahtar alanın hash sayısı hesaplanarak binary sayıya çevrilir. Bu hash sayısının son k dijiti sınır değerden küçükse son k+1 dijit, büyük yada eşitse son k dijit alınarak, buketdake yeri belirlenir. Bukette silinecek key'e ulaşılarak silinen kaydın yerine buketteki son anahtar konulur. Böylece arama olayı kolaylaşmış olur. Şekil (6.5.)

Böylece lineer hash metodu kayıt sayısı artarken yada azalırken birincil alan kolayca arttırılıp, eksilebilir. Dosyadaki kayıt sayısının birincil alan kapasitesine bölünmesiyle elde edilen yükleme faktörü sabit tutularak, tekrar düzenlemelerden kaçınılmış olur.



Şekil 6.5. Lineer Hash Metodu İle Buketlerden Alanları Silme.

BÖLÜM 7

SEYREK DEPOLAMADA BİLGİLERE ULAŞMA ZAMANINI AZALTMA TEKNİKLERİ:

Hash tablosuna bilgi girmek ve bu bilgilere erişmek için yeni bir metod geliştirilmiştir. Eğer bilgilerin bir kısmı birden fazla kez aranırsa metod kullanışlı olabilir. Eğer tablo tamamiyle dolu değilse, tablodaki bir bilgiye erişmek için yapılan araştırmaların sayısı diğer metodlarla mukayese edilebilecek kadar azdır.

Burada bilgilere erişmek için yapılan araştırma sayısı önce teorik olarak tasarlanmış ve daha sonra Monte Karlo deneyleriyle doğrulanmıştır.

Hash kodlama teknikleri tablolara bilgi girme ve onlara erişme zamanını minimize etmek için kullanılır. Compiler (derleyici) ve makine dillerinde, diskler gibi rasgele erişimli aygıtlara depo edilen, büyük dosyalarda da benzer teknikler kullanılır.

Amacımız sıralı aramaların (kayıta ulaşıp çağırma) etkin olabilmesi için bir metod tanımlamaktır.

Her bir bilginin, onu tanımlayan bir isim yada anahtar alan olduğunu farzedelim. k_1, $k_2 \ldots k_m$ gibi m tane anahtar alan uzunluğu $n \geq m$ olan h tablosunda $a(k_1)$, $a(k_2) \ldots$ $a(k_m)$ gibi adreslere yerleştirilirse ve k keyinin aranması istenildiğinde, problem k nın yerini, T tablosunda etkin olarak belirlemektir. Eğer k tablodaysa a(k) yı bulmaktır.

Farklı algoritmaların etkinliğini karşılaştırmak için

t deki elemanlara erişmek için yapılan araştırma sayısını dikkate almamız gerekir. Pratik uygulamalarda tablodaki bilgiler defalarca aranır. Daha açık örnek olarak belleğe ait kodlama tablosu yada reserve edilen kelimeler birkez oluşturulur ve yalnızca kayıtlara erişilirken kullanılır. Böylece çok önemli olan, tabloda bulunan anahtar alanları aramak için gerekli araştırmaların sayısını azaltmaktır. Tabloda olmayan anahtar alanları arama sayısı o kadar önemli değildir.

Bizim metodumuzun amacı, sıralı aramalar için gerekli araştırma sayısını indirgemek için anahtar alanları girerken, diğer genel metodlardan daha dikkatli olmaktır. Metodumuzu lineer bölüm metodunun bir modifikasyonu gibi göstermemize rağmen, aynı fikir diğer başka metodların madifikasyonundada kullanıldı. Örneğin; guadratik bölme metodundaki gibi.

Biz burada, bizim metodumuzdan biraz bahsettik. 2. kısımda detaylı olarak açıklayacağız. 3. kısında girişlerin yapılmasını ve onların aranması için gerekli araştırma sayısına göre metodumuzla diğer metodları kıyaslayacağız. 4. kısımda Monte Karlo deneylerinin sonuçlarını vereceğiz. 5. kısımda ise bazı kuramsal sonuçları ortaya çıkaracağız.

7.1. RİCHARD P.BRENT METODU

$n \geq 3$ olan bir asal sayı, T de m tane anahtar alan ihtiva eden n uzunluğunda bir tablo (örneğin bir dizi) olsun. Anahtar alanların nasıl tabloya yerleştirildiğini ve nasıl o anahtar alanlara ulaşıldığını anlatacağız.

Anahtarlar T tablosuna girerken, anahtarlarla birleşen

diğer değerlerde n uzunluğundaki başka bir tabloya yerleştirilirler. k anahtar alanının sıfırdan farklı bir tamsayı olduğunu düşünelim. (k=0 olamaz çünkü, sıfır T deki boş alanları göstermek için kullanılır.)

$0 \leq r < n$ ve $0 < q < n$ şartını sağlayan $r=R(k)$ ve $q=Q(k)$ lineer bölüm metodundaki gibi hesaplanır. Herhangi R ve Q pseudorandom fonksiyonları kullanılabilir. Bilgisayar için uygun bir bölme

$$R(k) = k \bmod n \qquad (1)$$

ve

$$Q(k) = (k \bmod (n-2))+1 \qquad (2)$$

(n tek olduğu için n-2 ile bölme, n-1 ile bölmekten oldukça iyidir. bölümde anlatılacağı gibi, k mod (n-1) in kalanı, k nın kalanıyla aynıdır.)
K nın T tablosunda aranıp bulunması için gerekli algoritma lineer bölüm metodundaki gibidir.

$s \geq 0$ için $hs = (r+sq) \bmod n$ ise (3)

bazı s ler için $T(h_0)$, $T(h_1)$.....'e bakılır ya a) $T(h_s)=k$ (yani anahtar aramadan sonra bulunur.) ya da b) $T(h_s)= 0$ $s \geq 0$ ve $h_s=h_0$) yani anahtar tabloda yoktur. N asal olduğu için eğer $s \geq 0$ ve $h_s=h_0$ ve $0 < q < n$ ise bütün tablo aranır.

T nin dolu olmadığını ve k yı kayıt yapmak istediğimizi farzedelim (k T de değil) k yı yukarıdaki algoritma ile arama işlemi $s \geq 0$ için $T(h_0) \neq 0, \ldots, T(h_{s-1}) \neq 0$ ve $T(h_s)=0$ ile biter

$1 \geq 0$ için $q_i = Q(T(h_i))$ (4)

ve $s \geq 1$ için

$h_{i,j} = (h_i + j\ q_i) \bmod n$ 'i tanımlayalım. (5)

bütün i,j ler arasından $T(h_{i,j})=0$ olacak şekilde i+j yi minimize eden i,j yi seçelim.
i nin minimize edilmesi durumunda iki olasılık vardır.

1) $1+j \geq s$: $T(h_s) \leftarrow k$ olacak şekilde, lineer bölüm metodunda olduğu gibi k yı T tablosuna yerleştir.

2) $i+j < s$: h_i de yer oluşturmak için h_i deki anahtar $h_{i,j}$ ye gönderilir. $T(h_i) \leftarrow k$ ve $T(h_{i,j}) \leftarrow T(h_i)$ k nın nasıl T ye gönderileceğinin (kayıt yapılacağının) belirlenmesi için s bilinir ve araştırma sayısı 1/2 (s(s-1)) den fazla değildir. ($h_{0,1} \cdots, h_{0,s-1}, h_{1,1} \cdots h_{1,s-2} \cdots h_{s-2,1}$ bölgelerinde)

1 durumu: $i+j \geq s$

s=3

$T(h_0) \neq 0 \longrightarrow T(h_{0,1}) \neq 0 \longrightarrow T(h_{0,2}) \neq 0$

$\downarrow$

$T(h_1) \neq 0 \longrightarrow T(h_{1,1}) \neq 0$

$\downarrow$

$T(h_2) \neq 0$

$\downarrow$

$T(h_3)=0$

2 durumu: s=3, i=0, j=2 $i+j < s$

$T(h_0) \neq 0 \longrightarrow T(h_{0,1}) \neq 0 \longrightarrow T(h_{0,2}) \neq 0$

$T(h_1) \neq 0$

↓

$T(h_2) \neq 0$

↓

$T(h_3) = 0$

Anahtarları h_i den $h_{i,j}$ ye göndermemizin nedeni: T deki $k_1, k_2, \ldots k_n$ anahtarlarını arayıp bulmak için gerekli araştırma sayısının toplamı

$$c = \sum_{1=1}^{m} p(k_i) \text{ dir.} \qquad (6)$$

Eğer anahtar alanların herbirinin bulunma olasılığı eşitse c mümkün olduğu kadar küçük tutulabilir. Bu durumda T deki herbir anahtarın aranıp bulunma olasılığı c/m dir. Böylece T ye yeni bir anahtar eklendiğinde, minimize edilen c içindeki bileşkesi artar. Yukarıdaki 1 durumunda p(k)=s+1 olur. Yani Δ=s+1 olur. 2 durumunda p(k)=i+1 ve p(T(h_i)),j ile artar. Yani i+j+1 olur. Böylece Δ yı mümkün olduğu kadar küçük tutmak için, eğer i+j $<$ s ise h_i deki veriyi $h_{i,j}$ ye göndermeliyiz.

7.2. Diğer Metodlarla Mukayese:

Seyrek depolama teknikleri içinde en iyisi lineer bölüm metodu ve quodratik bölüm metodudur. Bu iki metod birbirine çok benzer olduğu için biz Richard metodunu lineer bölüm metoduyla kıyaslayacağız.

Eğer n uzunluğundaki tabloda m tane kayıt varsa, yükleme faktörü α ; ($\alpha = \frac{m}{n+1}$) olarak tanımlanır. $\frac{1}{n}$ terimi ihmal

edilirse, giriş yapmak yada kayıtlara erişmek için gerekli inceleme sayısı tartışılırken bu yalnızca α nın bir fonksiyonudur. (q,r) çiftlerini bağımsız ve eşit olasılıklı olarak kabul ederek Q ve R iyi seçilmiş fonksiyonlar olsunlar. Tabloda olan bir anahtarı arayıp bulmak için beklenen araştırma sayısıyla yani $A(\alpha)$ ile, daha az ölçüde ise $B(\alpha)$ ile ilgileneceğiz, Bu da $(B(\alpha))$ Tabloda olmayan bir anahtarın arayıp bulunması için beklenen araştırma sayısıdır.

Lineer bölüm metodu için:

$$B(\alpha) = \frac{1}{1 - \alpha} \qquad (7)$$

$$A(\alpha) = \frac{1}{\alpha} \int_0^{\alpha} B(\beta)\,d\beta = \frac{1}{\alpha} \log \frac{1}{1-\alpha} \qquad (8)$$

metodumuzdaki $B(\alpha) = \frac{1}{1-\alpha}$ lineer bölüm metodundakinin aynıdır.

$A(\alpha)$ için ifade çıkarmak daha zordur. Yükleme faktörü β iken Tabloya bir anahtar girildiğinde $\Delta > d$ şartını sağlayacak şekilde Δ artacaktır.

$$T(h_0)\neq=0\ ,\ T(h_{0,1})\neq 0\ ,\ldots\ T(h_{0,d-1})\neq 0\ ,\ T(h_1)\neq 0,$$
$$\ldots T(h_{1,d-2})\neq 0,\ldots\ T(h_{d-1})\neq 0 \text{ dır.} \qquad (9)$$

herhangibir $T(h_{i,j})\neq 0$ olma olasığılı β olduğundan dolayı (9) $\beta^{d(d+1)/2}$ olasılıkla tutulur. O zaman Δ nın beklenen değeri

$$\sum_{d=0}^{\infty} \beta^{d(d+1)/2} \text{ olur.}$$

$$A(\alpha) \cong \alpha^{-1} \int_0^{\alpha} \sum_{d=0}^{\infty} \beta^{d(d+1)/2} \, d\beta$$

$$= \sum_{d=0}^{\infty} \alpha^{d(d+1)/2} \, (1+d(d+1)/2) \qquad (10)$$

$$= 1 + \frac{\alpha}{2} + \frac{\alpha^3}{4} + \frac{\alpha^6}{7} + \ldots$$

$P(Th(i,j)\neq 0)$ olasılıkları bağımsız olmadıkları için $A(\alpha)$ yaklaşımı tam anlamıyla doğru değildir. Bağımsızlığın olmaması anahtarları yukarıdaki 2 durumuna göre girmemize neden olur. 5. kısımda

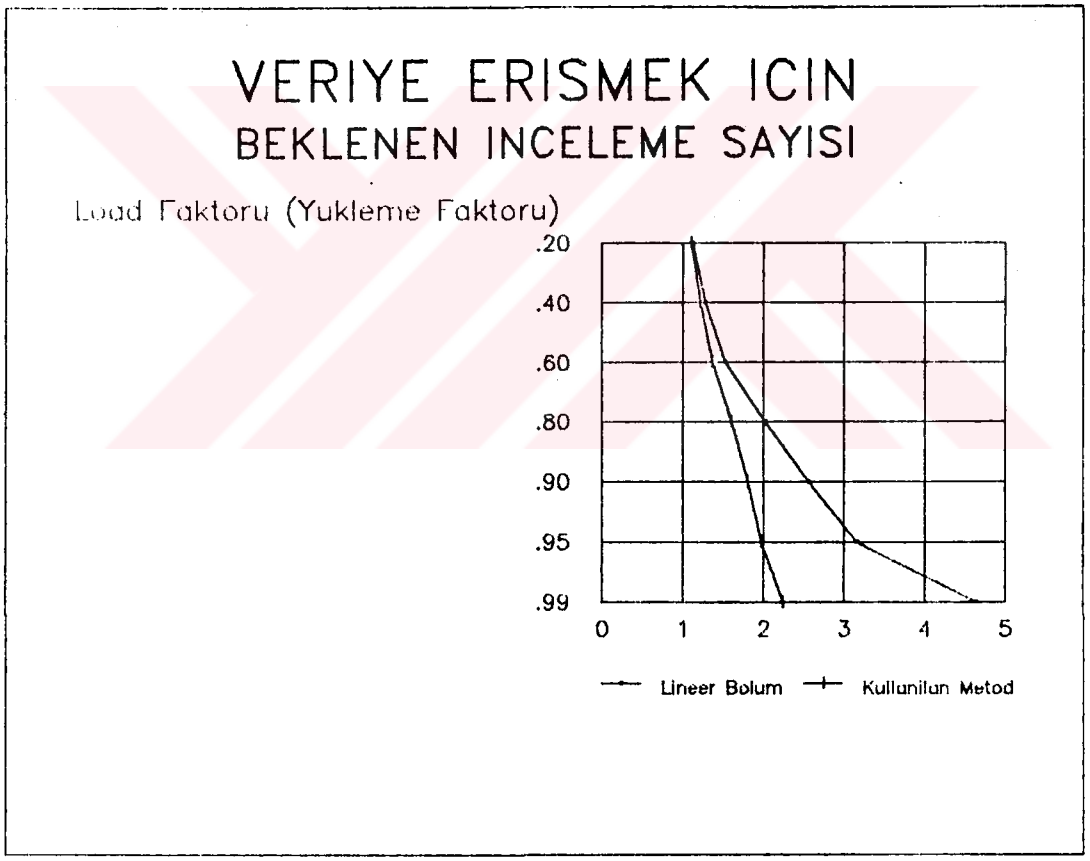
$$A(\alpha) = 1 + \alpha/2 + \alpha^3/4 + \alpha^4/15 - \alpha^5/18 + 2\alpha^6/15 + \ldots \qquad (11)$$

olduğunu göstereceğiz ve hesaplar gösterir ki $A(\alpha)$ daha küçültülebilir. Şekil 7.1. bizim metodumuz için ve lineer bölüm metudu ve direk zincirleme metodu için $A(\alpha)$ yı gösterir. Eğer α küçükse, farklı metodlar hemen hemen aynı etkinliktedir. Fakat eğer (yükleme faktörü) 1'e yakınsa metodumuz lineer bölüm metodundan hissedilir derecede daha etkindir. Direk zincirleme farklı uygulama alanlarına sahiptir ve diğer iki metotla mukayese edilemez. Direk zincirlemede bazı alanlar linklerle kaplıdır fakat bizin metodumuzda hiçbir linke gerek yoktur.

Gerekli alanın arttırımı linklerle değil, $A(\alpha)$ ve α yı azaltan tablo uzunluğunu arttırarak elde edilir. Uygulamada her iki metodda kullanılabilir. Eğer her bir anahtara bağlı olarak birleşen değerlerin sayısı çok fazla değilse, metodumuz lineer bölüm metodundan çok daha etkindir. Eğer tablo yaklaşık dolu ise metodumuz kesinlikle lineer bölüm metoduna tercih

edilebilir. Bu durumda lineer bölüm metodunda $A(\alpha)$ logn mertebesinde iken bizim metodumuzda

$$A(\alpha) < 2.5 \text{ dir.} \qquad (12)$$



Şekil 7.1. Beklenen inceleme Sayısı.

7.3. BAZI MONTEKARLO DENEYLERİ

Teorik sonuçları test etmek için uzunluğu (n) 4999 olan tablo pseudorandom anahtarlar kullanılarak doldurulur. Tabloyu doldururken bir giriş yapmak için gerekli ortalama sayıyı bulmak için yapılan tüm araştırma sayısını bir yerde topladık. Aynı şekilde her bir girişi bir kez bulmak için gerekli toplam araştırma sayısını da ihmal etmedik böylece bir giriş arayıp bulmak için gerekli ortalama sayıyıda bulmuş olduk.

Deney 1000 kez tekrar edildi ve sonuçlar Tablo 7.1. de özetlendi. $A(\alpha)$ metodumuz için bir bilgiyi arayıp bulmak için beklenen değerdir. $\hat{A}(\alpha)$ gözlenen anlamda alanı arayıp bulmak için araştıram sayısıdır. $\lambda(\alpha)$ lineer bölüm metodunda ki bir bilgiyi arayıp bulmak için beklenen değerdir.

$\lambda(\alpha)=(1/\alpha)\ \log\ (1/(1-\alpha))$ dır. $\hat{E}(\alpha)$ bizim metodumuzla bir giriş yapmak için gözlenen inceleme sayısıdır. Ve

$$\lambda(\alpha)=(\hat{E}(\alpha)-\lambda(\alpha))/\ (\lambda(\alpha)-\hat{A}(\alpha)) \qquad (13)$$

buradaki $\lambda(\alpha)$ bizim metodumuzu lineer bölüm metoduna üstün kılan herbir girişi iyi arayıp bulmak için gereken en küçük sayıdır.

Tablodan da görüleceği gibi $\hat{A}(\alpha)$ $A(\alpha)$ ya çok yakındır. Tablonun son koluna bakılırsa, bir bilgi,üç kez veya daha fazla arandığında metodumuzun lineer bölme metodundan daha etkin olduğu görülür. Bu birçok pratik uygulamada da böyledir.

Tablo 7.1. Monte Carlo deneylerinin sonuçları

α	$A(\alpha)$	$\tilde{A}(\alpha)$	$\lambda(\alpha)$	$\tilde{E}(\alpha)$	$\nu(\alpha)$
0.20	1.1021	1.1021	1.1157	1.1548	2.85
0.40	1.2178	1.2175	1.2771	1.4337	2.62
0.60	1.3672	1.3668	1.5272	1.9248	2.48
0.80	1.5994	1.5991	2.0118	1.9713	2.32
0.90	1.8023	1.8020	2.5584	4.2740	2.26
095	1.9724	1.9725	3.1534	5.8382	2.26
0.99	2.2421	2.2422	4.6517	10.3922	2.36

7.4. KURAMSAL SONUÇLAR

3. bölümde $A(\alpha)$ için bir yaklaşım çıkarmıştır. Burada çok daha iyi bir yaklaşımın nasıl bulunacağını kısaca tanımlıyacağız. n in büyük olduğunu farzedersek $\frac{1}{n}$ 'li terimleri ihmal edebiliriz. 2. bölümdeki notasyonda k tabloda h_s pozisyonunda bir anahtar ve $p(\alpha)$ $T(h_s+1)\neq 0 \ldots T(h_s+v)\neq 0$ için $v\geq 0$ olma olasılığı olsun. 10 formülünü çıkarmak için $p_v(\alpha)=\alpha^v$ olduğunu kabul ettik fakat bu $v\geq 1$ için tam doğru değildir. Eğer δ küçük δ_n tane yeni anahtar 2. bölümdeki algoritmaya göre tabloya girilirse.

$$(\alpha+\delta)\, p_v(\alpha+\delta) - \alpha p_v(\alpha) = \delta\alpha^v + (\delta/1-\alpha) \sum_{i=0}^{v-1} \alpha^{v-i} \,(p_i(\alpha)-p_{i+1}(\alpha))$$

$$+ \delta \sum_{i=0}^{\infty} \sum_{j=1}^{v} \alpha^{v-j}\, p_{i,j}(\alpha) + 0\,(\delta^2) \qquad (14)$$

buna da

$$p_{i,j} = \alpha^{i+j+1}\, p_{i+j} p_{i+j-1} \cdots p_{j+1} (p_{j-1}-p_j) p_{j-2} \cdots p_0 \qquad (15)$$

$P_{i,j}$ hi deki anahtar $h_{i,j}$ ye gönderildikten sonra giriş yapılma olasılığıdır. (14)'üncü formülün her iki tarafını δ ya böldükten son $\delta \longrightarrow 0$ için limitini alırsak

$$(d/d\alpha)(\alpha p_v) = (\alpha^v - \alpha p_v)/(1-\alpha) + \sum_{i=0}^{v-1} \alpha^{v-i} p_i + \sum_{j=1}^{v} \alpha^{v-j} \sum_{j=0}^{\infty} p_{i,j} \quad (16)$$

v=1,2... için $p_v(0)=0$ başlangıç şartı ile bu diferansiyel denklem sistemi $p_v(\alpha)$ fonksiyonlarını tanımlar. Eger $p_{i,j}$ ler ihmal edilirse denklem sistemi $p_v(\alpha) = \alpha^v$ gibi bir çözüme sahiptir. Bu lineer bölüm metodu için

$$A(\alpha) = -\frac{1}{\alpha} \int_0^{\infty} F(\beta)\, d\beta \quad \text{yı bulmak isteyelim} \qquad (17)$$

burada

$$F(\alpha)=1+ +d^2 p_1 + d^3 \beta_1 P_2 + \ldots. \qquad (18)$$

$F(\beta)$ yeni bir anahtar girildiğinde c deki Δ da beklenen artımdır. Eğer diferansiyel denklem sistemi sistemi numerik integrasyonla çözülürse

$$\left(\frac{d}{d\alpha}\right) A(\alpha) = (F(\alpha) - A(\alpha))/\alpha \qquad (19)$$

(16) ve (19) daki diferansiyel denklem sistemi, normal kuvvet serileri ile çözülürse

$$A(\alpha)=1+\alpha/2+\alpha^3/4+d^4/15-\alpha^5/18+2\alpha^6/15+9\alpha^2/80+293\alpha^8/5670-319\alpha^3/5600+..(20)$$

Bu α 1'e yaklaşmadıkça $A(\alpha)$ ya yaklaşma şartını verir. Eğer α bire yaklaşırsa büyük sayıdaki terimleri içerecektir. Böylece sistemin nümerik integrasyonundan daha iyi gibi görünür.

$A(\alpha)$ nın $\alpha=1$ deki dik teğetinden dolayı numerik zorluklardan kurtulmak için

$$\alpha=1-\varepsilon \tag{21}$$

değişikliğini yapmağa değer. $A_\varepsilon(1-\xi^2)$ fonksiyonu ε {0,1} için iyidir. $\varepsilon=0$ nümerik extrapolasyon ile

$$\lim_{\alpha \to 1^-} A(\alpha) \cong 2,4941 \tag{22}$$

buluruz. Böylece (12) eşitsizliği bütün α için sağlanır.

7.5. SONUÇ

Metodumuzun lineer bölüm metoduna üstün olan taraflarını tartıştık ve birçok bilginin birden fazla anahtarın aranıp bulunması şartıyla, tablo yaklaşık (hemen, hemen) dolu ise bu metoddan epey farklı olduğunu söyledik.

Buraya kadar bazı bilgilerin silindiğinden (silinmiş olabileceğinden) hiç söz etmedik bu gibi durumlarda yukarıda çıkarılan sonuçlar geçersizdir.

Bilgiler silindiğinde, diğer bilgilere erişebilme durumuna dikkat edilmelidir. Arama sayısını arttırsa bile, çözüm silinen bilgiyi belirli bir anahtar alanla göstermektir.

Eğer tabloda fazla sayıda silinmiş kayıt varsa anahtar alanları ararken yada girerken yapılan hesaplamaların sayısı artacaktır.

Yukarıda tanımlanan algoritma kompütürlerin yüksek hızlı, rasgele erişimli belleklerinde bir tablo (hash tablosu) kullanımı için uygundur. Eğer tablo disk gibi bir aygıta yüklenirse bazı modifikasyonlar gerekebilir.

Örneğin: Birkaç anahtarla birleşen bilgilerin bir disk trackine depo edildiğini farzedelim. Bir track üzerinde bazı hesaplamalar yaptıktan sonra, aynı track üzerinde başka bir hesaplama, diğer başka bir track üzerinde yapılacak hesaplamalardan (aramalarda) daha kolay olacaktır. Bir track üzerine kayıt yapmak istediğimizde collision durumunda başka trackler çizerinde hesaplamalar yapmadan önce bu track üzerinde kayıt yapmak denemeye değerdir.

Takip edilen bu yolla beklenen sayı artacaktır, fakat farklı tracklerde arama sayısı azalacaktır. Numaralanmış hafıza kullanımında benzer düşünceler uygulanır.

BÖLÜM 8

QUICKSORT SIRALAMA METODU

Quicksort C.A.R Hoare tarafından bulunmuştur. Şimdiye kadar kullanılan sıralama algoritmalarının içinde en iyisidir.

Bir dizi Quicksort kullanılarak sıralanıyorsa, dizinin içinde bir değer seçilir (comparand) ve bu seçilen değerden büyük veya eşit olanlar bir tarafta ve seçilen değerden küçük olanlar bir tarafta olmak üzere, dizi iki parçaya bölünür. Dizinin tamamı sıralanana kadar bu işleme devam edilir. Örneğin: verilen dizi fedacb ise ve seçilen değer d ise Quicksort ilk kullanıldığında bcadef haline gelir. Bu işlem her bir parça için terkarlanır. (bca ve def için)

Mukayese edilerek değerin seçilmesi için iki yol vardır. Birincisi rasgele seçmektir, ikincisi ise dizi elemanlarının ortasındaki ((ilk+son)/2) elemanının seçilmesidir. Mukayese etme sayısı nlogn ve yapılan değiştirme sayısı ise n/6 logn dir. Shell sıralama ve bubble sıralama ile mukayese edildiğinde, bu sayının daha küçük olduğu görülür.

Sıralanması istenen dizinin 100 tane elemanı olduğunda log 100=2 olduğu için mukayese sayısı n.logn yani 100.2 dir. Buble sort da bu sayının 990 olduğu düşünülürse bu sayı gayet iyidir. [3]

BÖLÜM 9

VERİTAB PAKET PROGRAMININ YAPISI

Bu kısımda programda kullanılan alt programlar ve program hakkında bilgi verilecektir.

Main ( ): Ana program yapısını oluşturur. Dosoku alt programı çağrılarak dosyadaki kayıt sayısı belirlenir. Kayıt sayısı sıfır ise DOSYADA KAYIT YOK mesajı verir. Kayıt dosyasına göre buket sayısı hesaplanır, anahtarları buketlere yerleştiren alt programlar çağırılır. Daha sonra menü alt programı ile ana menüyü e-rana getirir.

Kayıt-girişi ( ): Bu alt programda 10 tane dosya kullanılır. Ana dosyaya her bir kayıt için sicil numarası, adı soyadı, telefon numarası, adresi, doğum yeri ve doğum tarihi bilgileri ana dosyaya yazdırılır. Kayıt işlemi bittikten sonra dosyadaki herbir alan için quick-sort algoritması izlenerek, alanlar sıralanır ve 6 ayrı dosyaya yazılırlar bunlar program paketinin indeks dosyalarıdır. Sıralama olayı tamamlandıktan sonra ilk ( ) alt programı çağrılarak lineer hash metodu kullanılır.

Kayıt-okuma ( ) : Dosyadaki kayıtların görüntülenmesini sağlar. Seçeneğe göre kayıtlar tek tek yada sayfa, sayfa gözden geçirilir.

Sıralama( ) : Sıralama menüsünü oluşturan alt programdır. Sicil numarasına, isme, telefon numarasına, adrese doğum yerine ve tarihine göre ok tuşlarıyla seçenek alınarak, sıralanmış kayıtlar görüntülenir.

Has (snn): Has alt programının son parametresi, alt programlardan has alt programına gönderilen anahtar alandır. snn in herbir karakterinin ASCII kodlarının toplamı alınarak çağıran programa gönderilir.

Hes(aa,bb): burada aa parametresi, anahtar alanının ASCII kodlarının toplamıdır. aa hash sayısı bu alt proğramda Linary (ikili) sayı sistemine çevrilir. bb ise binary sayının uzunluğudur. Hash sayısı ikili sayı sistemine çevrilirken bitwise operatörlerinden yararlanılmıştır. Bitwise operatörleri C yi Assembler seviyesine indirirler. Bu alt programda shift left operatörü kullanılmıştır.

Ayır(rt,xy): Hes alt programında bulunan hash sayısının ikili sayı sistemine çevrilmesinden sonra, hesep (ma) alt programında hesaplanan dijit sayısı kadar sondan xy dijit alınır ve vop adlı değişkene atanarak çağıran programa gönderilir.

Mu(vop,bh): bh parametresi sınır değerdir. Daha önceki alt programda binary sayıya çevrilen hash sayısının son vop dijiti, sınır değerle mukayese edilir. Sınır değerden küçükse son vop+1 dijit, büyük yada eşitse son vop kadar dijit alınıp, tamsayıya çevrilir. Bu tam sayı 0 ile Ma-1 (buket sayısı-1) arasındadır. Anahtar alanların kaçıncı bukete yerleştirileceğini belirleyen değeri çağıran programa gönderir.

qyer(q_2,q_1): Programda ok tuşları ile yukarı, aşağı hareket etmemizi sağlar. Return tuşuna basarak bu alt programdan çıkılır. q_2 ve q_1 parametreleri ekrandaki cursor koordinatlarıdır.

Kayıt-silme( ): Ana dosyadan kayıtların silinmesini sağlayan alt programdır. Bu alt progrmada kayıtlar dosyadan silinirken del, hes, has, ayır, mu alt programları kullanılarak, anahtar

alanlar bellekten, lineer hash metodu kullanılarak silinirler. Silen isimli yeni bir dosyaya da silinen kayıtlar yazdırılırlar.

Stuct bucket del (sir,cop): Lineer hash metoduna göre, anahtar alanı bellekteki buketlerden silen alt programdır. Cop parametresi, silinmek istenen anahtar alanın bulunduğu structure (yapı)dır, sir ise bu anahtar alanın bulunduğu buket dir. Silinmek istenen anahtar alan bukette yok ise ARADIĞINIZ KAYIT YOK mesajı alınır.

Struct bucket ara (sir, cop): Bellekte lineer hash metoduna göre arama yapan alt programdır. Aranan anahtar alan bulunduğinda buketten kayıt numarası alnarak, başlık ( ) alt programı çağrılarak aranan kayıt ekranda görülür. Aksi durumda ARADIĞINIZ KAYIT YOK mesajı alınır.

Hesap (ma): Program ilk çağrıldığında hesaplanan buket sayısı ma parametresidir. Bu alt programda $2^k \leqq Ma < 2^{k+1}$ şartını sağlayan dijit sayısı hesaplanır ve çağıran programa gönderilir.

İlk ( ): Programda buketleri oluşturamak için bellekte buket sayısı kadar yer ayırır.

Struct bucket * ekle (xx,ac): Sicil numaralarını buketlere yerleştirir. Bu alt programda ac parametresi bukete yerleştirilecek sicil numarası, xx ise yerleştirileceği bukettir. Burada buket yapısı alanlar ve aynı alanlardan oluşan buket tipindeki linkden oluşmuştur.

Struct ket * tı (xx1,ac1): Arama menüsünde isme göre arama yapılabileceğinden ikinci alan olarak isim kullanılmıştır. Sicil numarasında olduğu gibi, her bir isim için has alt programıyla hash sayısı hesaplandıktan sonra hes, ayır, mu alt

programları kullanılarak isimler buketlere yerleştirilirler. Burada xx1 parametresi ismin gönderileceği buket, ac1 parametresi ise bu bukete yerleştirilecek alan isimdir.

Değiştirme ( ): Bu alt programda kayıt değiştirme menüsü ekrana gelir. Kayıtlarla ilgili değişiklikler qyer alt programı kullanılarak, aşağı , yukarı tuşları yardımıyle yapılır. Return tuşuna basılarak qyer alt programından çıkılır. Cursor'un satır konumuna göre belirlenen alanlarda değişiklik yapılır ve dosyaya yazdırılır. Değiştirme menüsünden ESC tuşu yardımı ile çıkılır.

Arama ( ): Bu alt programa gelindiğinde, Ekranda Arama menüsü görüntülenir. Bu menüde qyer alt programı kullanılarak yukar aşağı tuşları kullanılarak seçenek alınır.

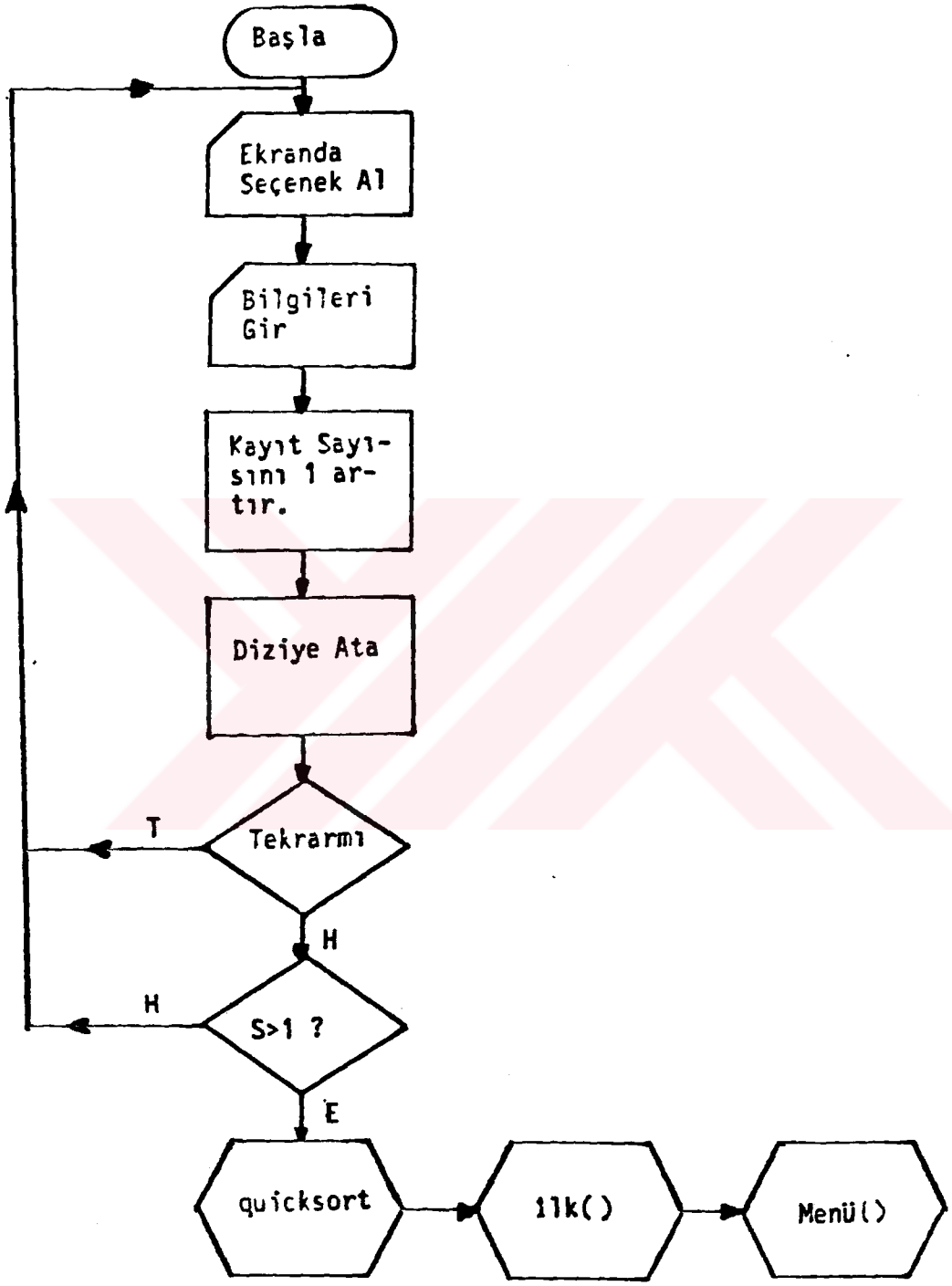
Menu ( ): Programdaki diğer menülere geçmek için gereken alt programların seçilmesini sağlayan ara menüsü ekrana getirilerek alt programları çağıran alt programdır. Programın herhangi bir yerinde ana menüye dönülmek istendiğinde menü( ) çağrılır.

Indeks ( ): İndeks menüsünü ekrana getirir. İndeks dosyalarından kayıt numaraları alınarak, ana dosyadaki kayıtlar ekrana getirilir. Veri alanları quick-sort algoritması ile sort, snt, dog, sırt, 54, nor isimli alt programlarda sıralanmışlardır.

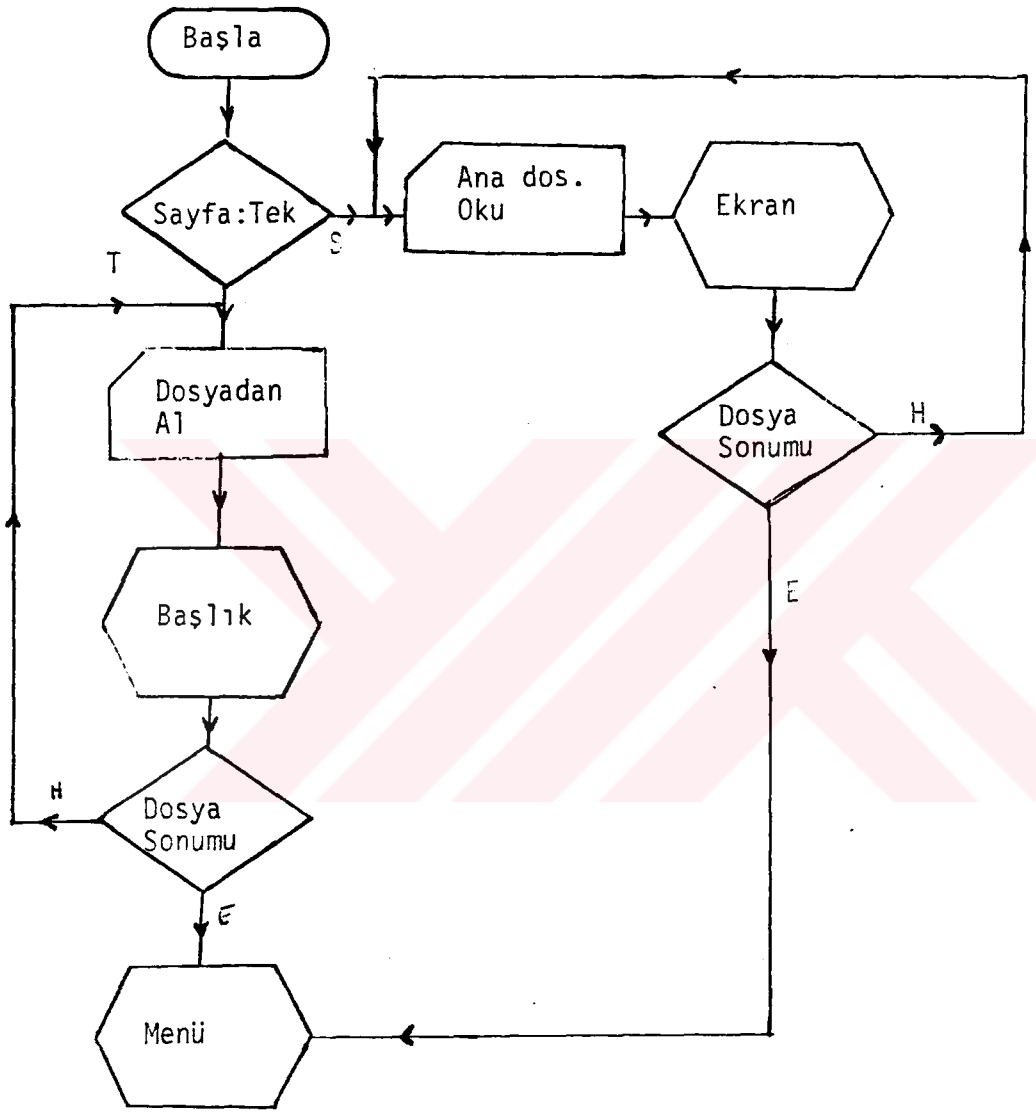
Dosya-ac ( ): İndeks dosyalarının, Ana dosyanın ve silen isimli dosyanın açılmalarını sağlar. Programda aynı anda 10 tane dosya açılır.

Dosya-isimleri ( ): Dosyalama menüsünü ekrana getirir. Dosya-ac( ), liste ( ), dossil ( ) alt programlarını çağırarak dosyaların yaratılmasını, listelenmesini ve silinmesini gerçekleştirir.

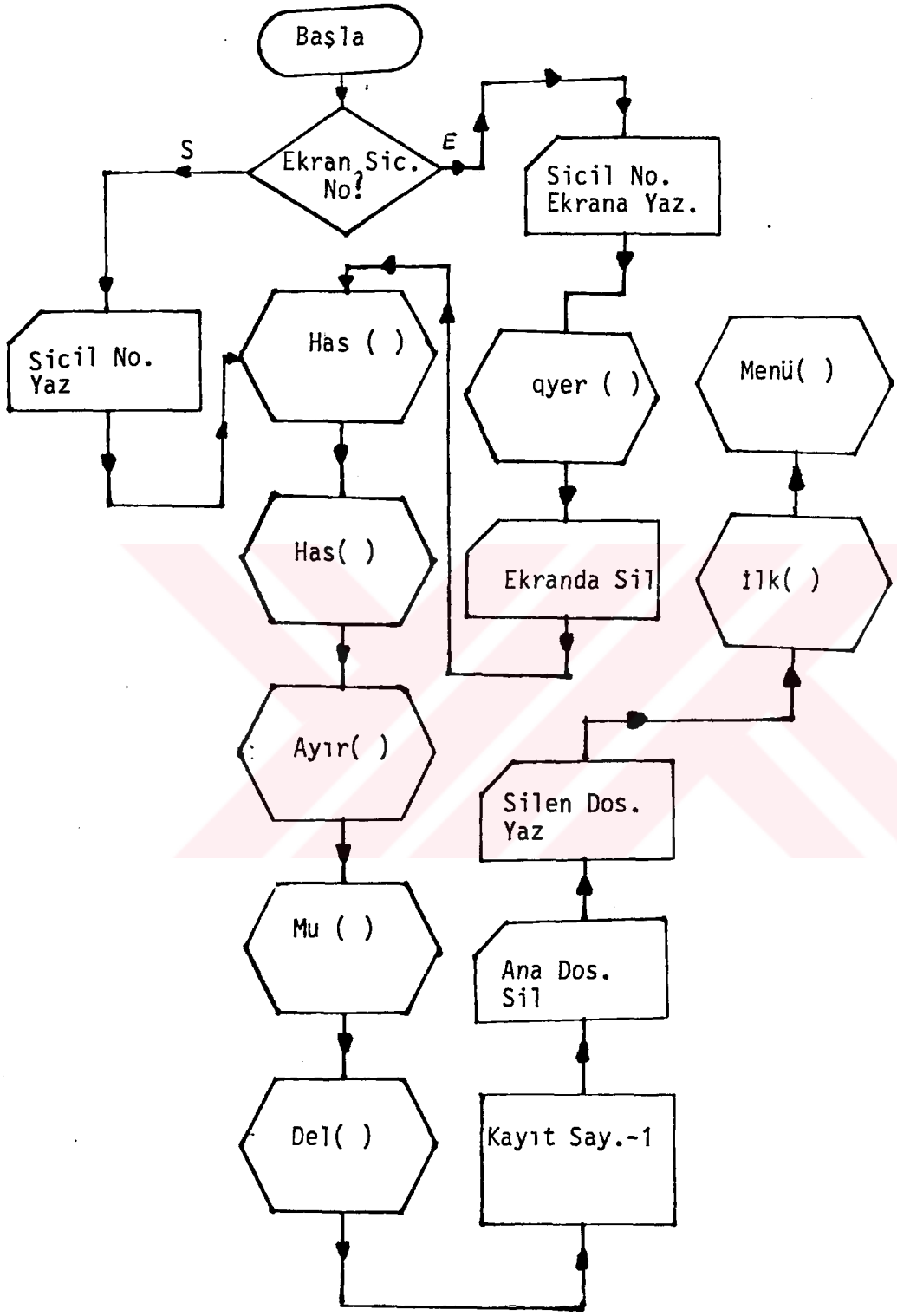
Undel ( ) : Silinen kayıtlar içinde, geri alınması (dosyaya yeniden eklenmesi) istenen kayıtları dosyaya yazar. Kayıtlar silinirken silen adlı dosyaya yazılırlar. Geri alınma durumunda bu dosyadan alınarak ana dosyanın sonuna eklenirler.



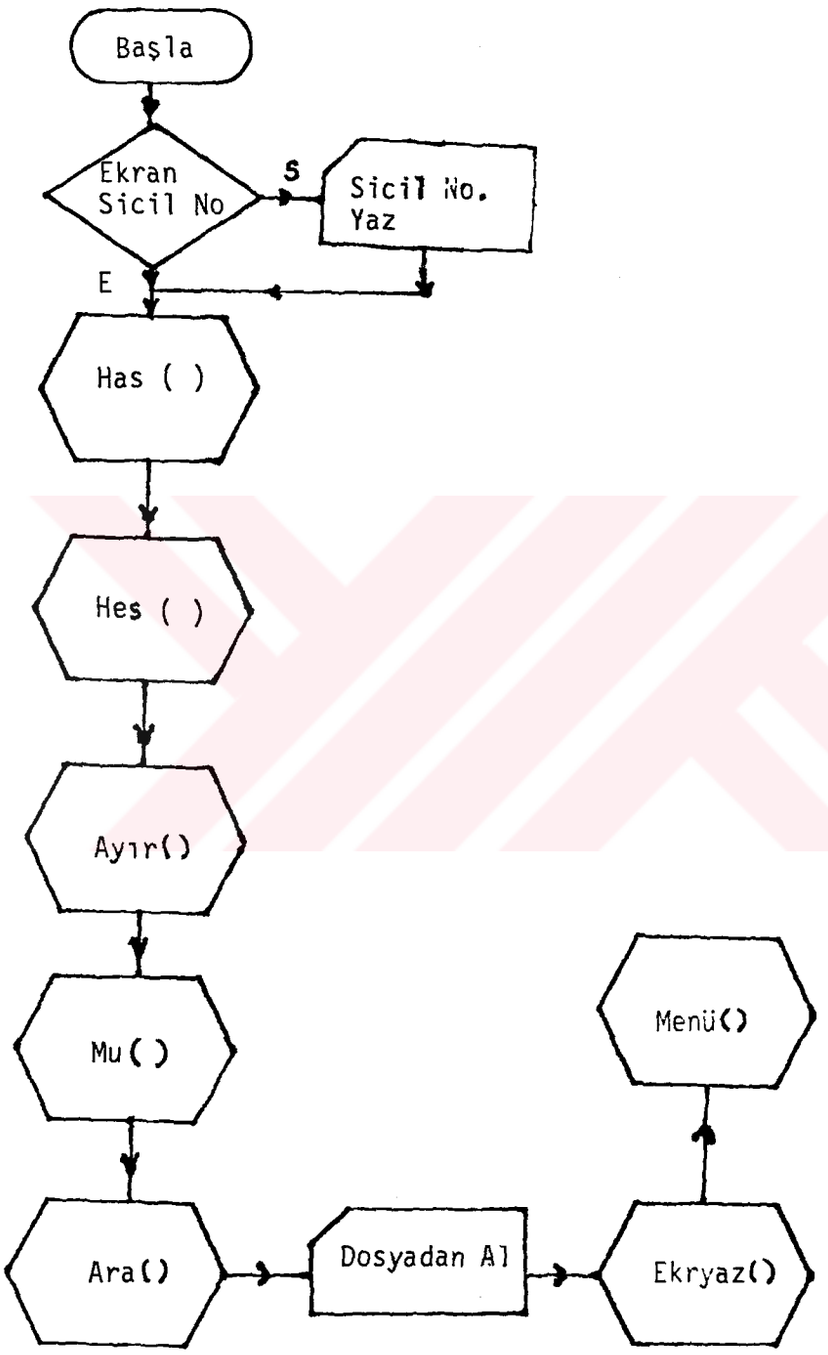
Şekil 9.1. Kayıt Girişi Alt Programı



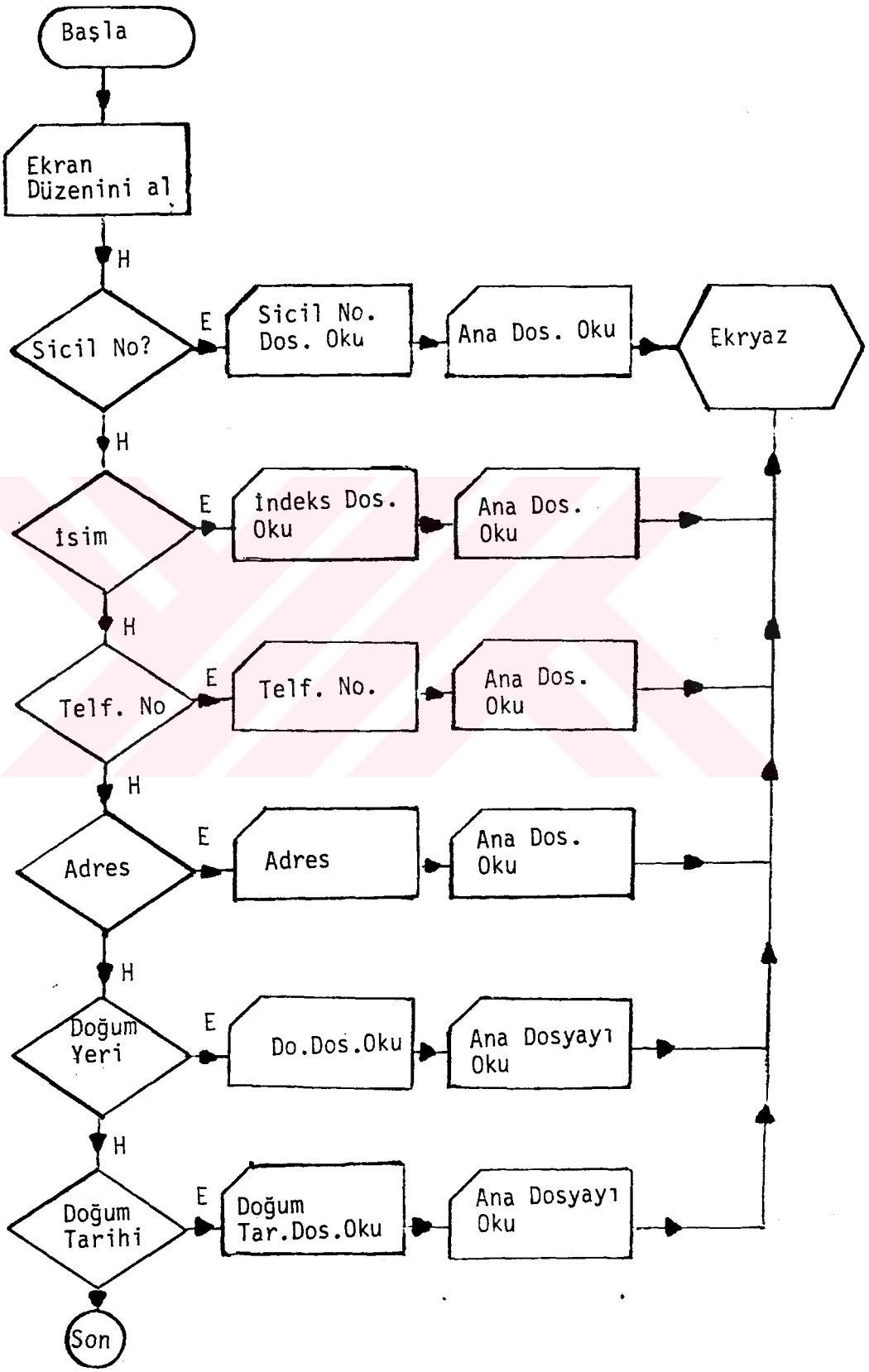
Şekil 9.2. Kayıt Okuma Alt Program.



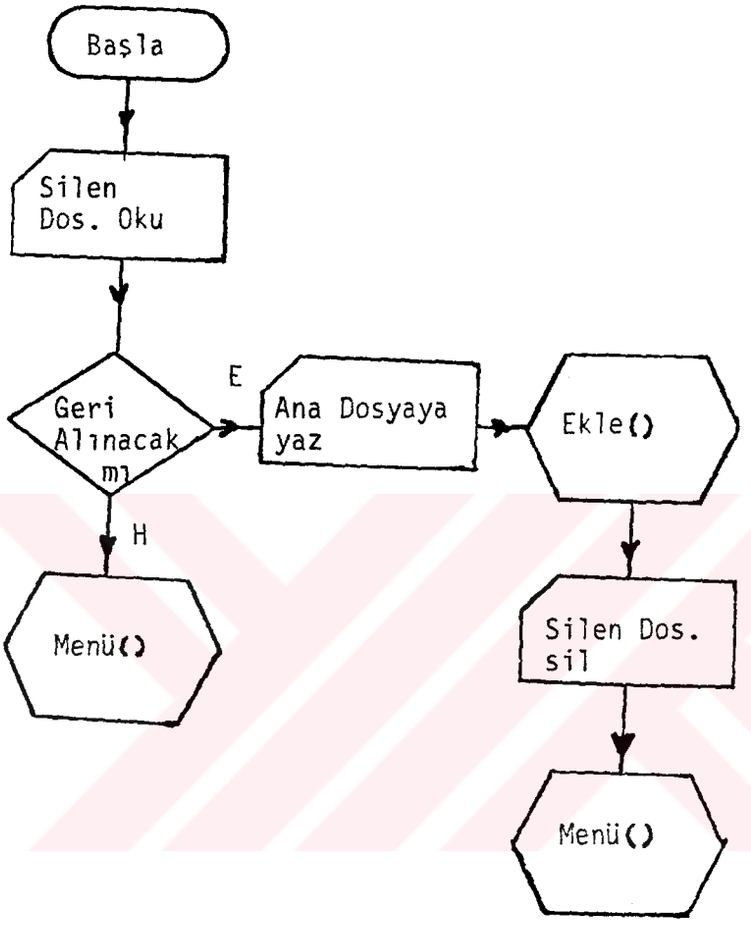
Şekil 9.3. Kayıt Silme Alt Programı



Şekil 9.4. Kayıt Arama Alt Program



Şekil 9.5. Kayıt Sıralama Alt Programı



Şekil 9.6. Undel Alt Programı.

BÖLÜM 10

VERİTAB PAKET PROGRAMININ KULLANILIŞI

Veritab paket programı PC ler için geliştirilmiş olup işletim sistemi MSDOS işletim sistemidir. Turbo C programlama dilinde yazılmıştır. C>VERİTAB yazılarak çalıştırılır ve aşağıdaki menü ekrana gelir.

Şekil 10.1. Programın ana menüsü

Şekil 10.1. de görülen menü programın ana menüsüdür. Ekrandaki görünümünde diğer menülere geçmek için basılacak harfler büyük harfle ve parlak olarak değişik renklerle ekrana yazılmıştır.

10.1. Çalışma Esnasında Kullanılabilecek Tuşlar ve Ekranlar

Programın çalışması esnasında, ekranda görülen menülerin altında bazı işlemleri gerçekleştirecek tuşlar vardır. Ana menüden F1 tuşuna basılarak çıkılır. Diğer menülerde, yukarı aşağı tuşları kullanılarak, üst ve alt menülere geçilebilir. D: Dosyaya kayıt yapabilmek için, önce dosyaların yaratılması gerekir. D toşuna basılarak ana menüden dosyalama menüsüne geçilir. Bu menüde yine D tuşu yardımıyla, programda kullanılacak bütün dosyalar açılır. L tuşuna basarak, C directöry'sinde bulunan bütün dosyaların listesi alınır. Şekil 10.2.

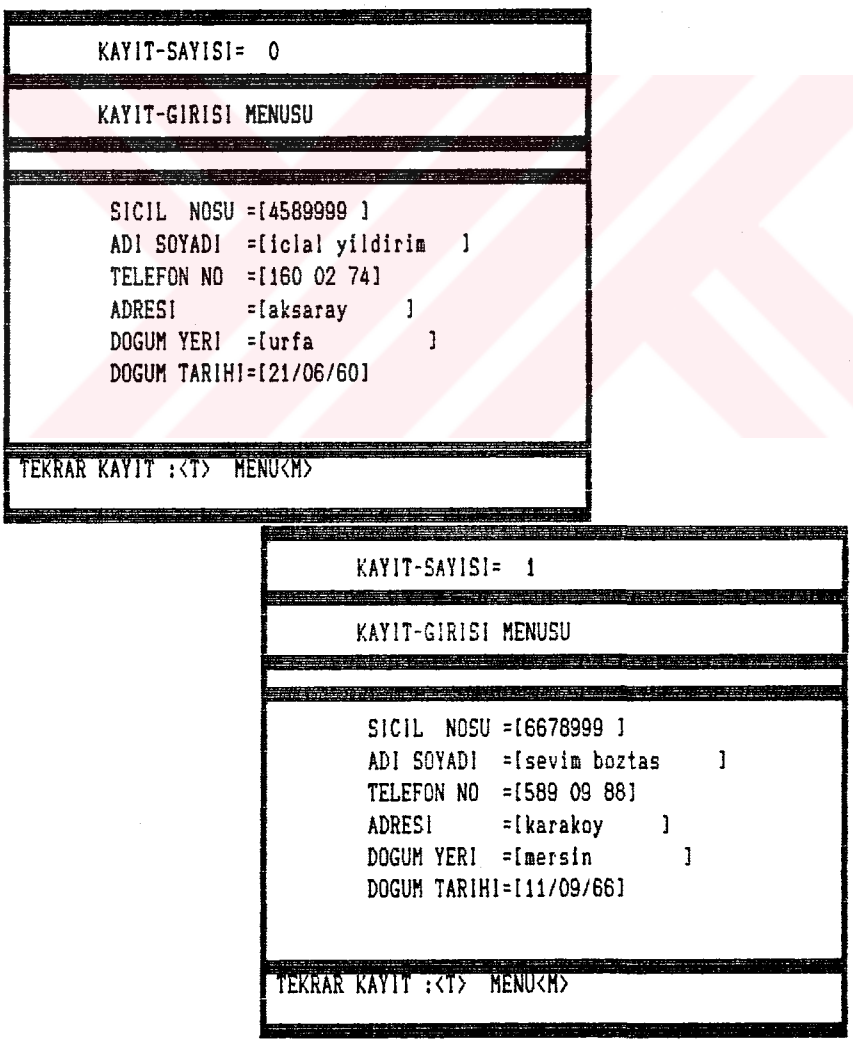
```
                    DOSYALAR OKUNUYOR.....

TOUCH.COM           SANS.CHR            MCOMMAND.C
TCC.EXE             TRIP.CHR            MCPARSER.C
CPP.EXE             BUILD-CO.BAT        MCUTIL.C
TCINST.EXE          CO.ASM              TURBOC.CFG
TLINK.EXE           EMUVARS.ASI         REDUCE.OBJ
HELPME!.DOC         MAIN.C              NEYA.BAK
TLIB.EXE            RULES.ASI           HODRI.EXE
OBJXREF.COM         SETARGV.ASM         HODRI.BAK
GETOPT.C            SETENVP.ASM         LAI.BAK
HELLO.C             WILDARGS.OBJ        OUR.OBJ
MATHERR.C           CBAR.C              SAYI
SSIGNAL.C           CPASDEMO.C          SI
CINSTXFR.EXE        CPASDEMO.PAS        BAL
bir tusa basiniz... bir tusa basiniz.... bir tusa basiniz....
```

Şekil 10.2. Directorydeki dosyaların listesi

S tuşuna basarak, programda kullanılan dosyalar sildirilir.

K: Bu tuşa basarak, her bir kayıt için sicil numarası, adı soyadı, telefon numarası, adresi, doğum yeri ve doğum tarihi bilgileri dosyaya yazılır. Telefon numarası ve doğum tarihi girilirken, istenilen formatda yazılması için ekranda bazı mesajlar görülür. Kayıt işlemine devam etmek için T tuşuna, Ana Menüye dönmek için ise M tuşuna basılır.



Şekil 10.3. Kayıt giriş menüsü

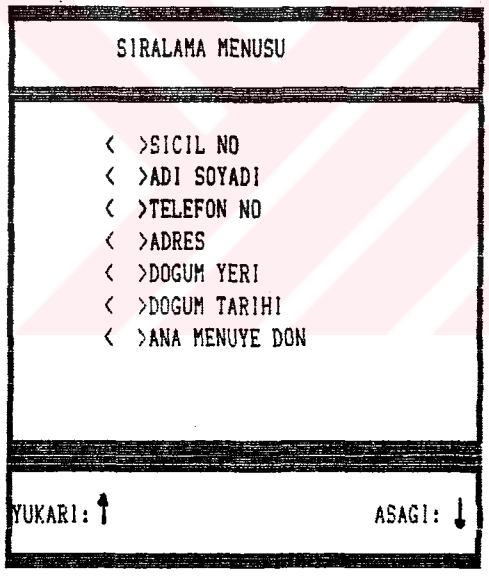
G: Ana menüden G seçeneği alınarak, dosyadaki bütün kayıtlar gözden geçirilir. Dosyadaki kayıtlar yapılan kayıt sayısına göre ekrana dökülürler.

Kayıtları sayfa sayfa görmek istersek S, tek tek görmek istiyorsak T seçeneğini seçiyoruz.

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
2	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66
3	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
4	nurel dogan	7188999	157 88 99	capa	ankara	14/07/71
5	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
6	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
7	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61
8	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
9	salih celik	7651999	160 02 75	atakoy	kirikkale	15/02/76
10	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
11	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
12	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
13	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
14	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61

Şekil 10.4. Gözlem menüsü

I: Silme ile sıralama kelimelerinin baş harfleri aynı olduğundan sıralama menüsüne geçmek için I harfi büyük harflerle ve parlak renkli olarak yazılmıştır. I tuşuna basılarak, veri alanları (alfabetik, nümerik ve tarihsel alanlar) küçükten büyüğe doğrultusunda saralanır. Sıralama menüsünde yine yukarı ↑ aşağı ↓ tuşları kullanılarak, istenilen alana göre sıralanmış kayıtlar görülür.



Şekil 10.5. Sıralama menüsü

Ksa	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOGUM YERI	DOG.TAR .
1	3088999	ali beyaz	525 02 15	uskudar	izmit	09/09/30
2	3290599	hale gurler	172 15 18	kadikoy	bursa	09/08/32
3	3567999	nuri deniz	157 88 88	bakirkoy	giresun	12/07/35
4	4589999	iclal yildirim	160 02 74	aksaray	urfa	21/06/60
5	4800999	sabahat bilgin	588 55 66	yesilyurt	edirne	19/10/48
6	6123499	can kavuzlu	518 99 99	cerrahpasa	tokat	29/02/61
7	6139999	nimet altinel	589 05 66	topkapi	amasya	14/03/61
8	6678999	sevim boztas	589 09 88	karakoy	mersin	11/09/66
9	7188999	nurel dogan	157 88 99	capa	ankara	14/07/71
10	7569999	jale ozdemir	171 14 15	gayrettepe	artvin	15/02/75
11	7651999	salih celik	160 02 75	atakoy	kirikkale	15/02/76
12	8199799	kemal topuz	175 66 66	samatya	izmir	17/04/81
13	8896999	ozlem yildirim	589 61 06	findikzade	istanbul	12/10/88
14	8991099	kamil soydan	176 35 88	ayazaga	giresun	18/10/89

DEVAM<D>----MENU<M>

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
2	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61
3	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
4	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
5	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
6	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
7	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
8	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61
9	nurel dogan	7188999	157 88 99	capa	ankara	14/07/71
10	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
11	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
12	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
13	salih celik	7651999	160 02 75	atakoy	kirikkale	15/02/76
14	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66

Şekil 10.6. Sicil No ve adı soyadı alanlarına göre sıralı kayıtlar.

Ksa	TEL NO	SIC.NO	ADI SOYADI	ADRES	DOGUM YERI	DOG.TAR.
1	157 88 88	3567999	nuri deniz	bakirkoy	giresun	12/07/35
2	157 88 99	7188999	nurel dogan	capa	ankara	14/07/71
3	160 02 74	4589999	iclal yildirim	aksaray	urfa	21/06/60
4	160 02 75	7651999	salih celik	atakoy	kirikkale	15/02/76
5	171 14 15	7569999	jale ozdemir	gayrettepe	artvin	15/02/75
6	172 15 18	3290599	hale gurler	kadikoy	bursa	09/08/32
7	175 66 66	8199799	kemal topuz	samatya	izmir	17/04/81
8	176 35 88	8991099	kamil soydan	ayazaga	giresun	18/10/89
9	518 99 99	6123499	can kavuzlu	cerrahpasa	tokat	29/02/61
10	525 02 15	3088999	ali beyaz	uskudar	izmit	09/09/30
11	588 55 66	4800999	sabahat bilgin	yesilyurt	edirne	19/10/48
12	589 05 66	6139999	nimet altinel	topkapi	amasya	14/03/61
13	589 09 88	6678999	sevim boztas	karakoy	mersin	11/09/66
14	589 61 06	8896999	ozlem yildirim	findikzade	istanbul	12/10/88

DEVAM<D>----MENU<M>

Ksa	ADRES	SIC.NO	ADI SOYADI	TEL NO	DOGUM YERI	DOG.TAR.
1	aksaray	4589999	iclal yildirim	160 02 74	urfa	21/06/60
2	atakoy	7651999	salih celik	160 02 75	kirikkale	15/02/76
3	ayazaga	8991099	kamil soydan	176 35 88	giresun	18/10/89
4	bakirkoy	3567999	nuri deniz	157 88 88	giresun	12/07/35
5	capa	7188999	nurel dogan	157 88 99	ankara	14/07/71
6	cerrahpasa	6123499	can kavuzlu	518 99 99	tokat	29/02/61
7	findikzade	8896999	ozlem yildirim	589 61 06	istanbul	12/10/88
8	gayrettepe	7569999	jale ozdemir	171 14 15	artvin	15/02/75
9	kadikoy	3290599	hale gurler	172 15 18	bursa	09/08/32
10	karakoy	6678999	sevim boztas	589 09 88	mersin	11/09/66
11	samatya	8199799	kemal topuz	175 66 66	izmir	17/04/81
12	topkapi	6139999	nimet altinel	589 05 66	amasya	14/03/61
13	uskudar	3088999	ali beyaz	525 02 15	izmit	09/09/30
14	yesilyurt	4800999	sabahat bilgin	588 55 66	edirne	19/10/48

Şekil 10.7. Telefon numarası ve adrese göre sıralı kayıtlar.

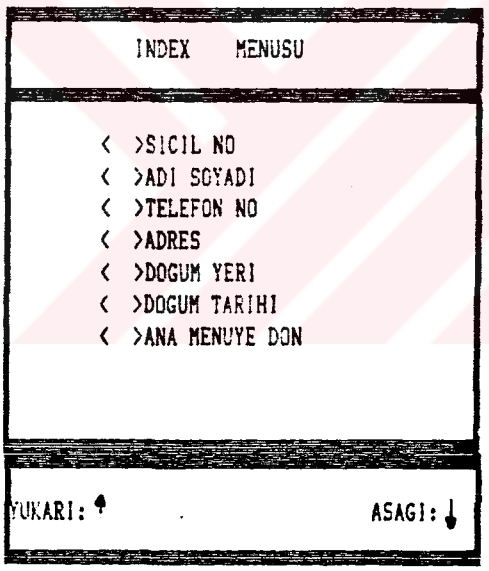
Ksa	DOGUM YERI	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOG.TAR.
1	anasya	6139999	nimet altinel	589 05 66	topkapi	14/03/61
2	ankara	7188999	nurel dogan	157 88 99	capa	14/07/71
3	artvin	7569999	jale ozdemir	171 14 15	gayrettepe	15/02/75
4	bursa	3290599	hale gurler	172 15 18	kadikoy	09/08/32
5	edirne	4800999	sabahat bilgin	588 55 66	yesilyurt	19/10/48
6	giresun	8991099	kamil soydan	176 35 88	ayazaga	18/10/89
7	giresun	3567999	nuri deniz	157 88 88	bakirkoy	12/07/35
8	istanbul	8896999	ozlem yildirim	589 61 06	findikzade	12/10/88
9	izmir	8199799	kemal topuz	175 66 66	samatya	17/04/81
10	izmit	3088999	ali beyaz	525 02 15	uskudar	09/09/30
11	kirikkale	7651999	salih celik	160 02 75	atakoy	15/02/76
12	mersin	6678999	sevim boztas	589 09 88	karakoy	11/09/66
13	tokat	6123499	can kavuzlu	518 99 99	cerrahpasa	29/02/61
14	urfa	4589999	iclal yildirim	160 02 74	aksaray	21/06/60

DEVAM<D>----MENU<M>

Ksa	DOG.TAR.	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOGUM YERI
1	18/10/89	8991099	kamil soydan	176 35 88	ayazaga	giresun
2	12/10/88	8896999	ozlem yildirim	589 61 06	findikzade	istanbul
3	17/04/81	8199799	kemal topuz	175 66 66	samatya	izmir
4	15/02/76	7651999	salih celik	160 02 75	atakoy	kirikkale
5	15/02/75	7569999	jale ozdemir	171 14 15	gayrettepe	artvin
6	14/07/71	7188999	nurel dogan	157 88 99	capa	ankara
7	11/09/66	6678999	sevim boztas	589 09 88	karakoy	mersin
8	14/03/61	6139999	nimet altinel	589 05 66	topkapi	amasya
9	29/02/61	6123499	can kavuzlu	518 99 99	cerrahpasa	tokat
10	21/06/60	4589999	iclal yildirim	160 02 74	aksaray	urfa
11	19/10/48	4800999	sabahat bilgin	588 55 66	yesilyurt	edirne
12	12/07/35	3567999	nuri deniz	157 88 88	bakirkoy	giresun
13	09/08/32	3290599	hale gurler	172 15 18	kadikoy	bursa
14	09/09/30	3088999	ali beyaz	525 02 15	uskudar	izmit

Şekil 10.8. Doğum yeri ve doğum tarihine göre sıralı kayıtlar

X: Bu tuşa basılarak programdaki, indeks menüsüne ulaşılır. Herhangi bir alana göre indeksli kayıtların görülebilmesi için yukarı ↑, aşağı ↓ tuşları kullanılır. Seçenek alınarak dosyadaki seçilen alanlar için indeksli kayıtlar ekranda görüntülenir. İndeks menüsünden seçenek alınmasına devam etmek için, D tuşuna ana menüye dönmek için M tuşuna basılır.



Şekil 10.9. İndeks menüsü (Yukarı, aşağı tuşları kullanılarık seçerek alınır.)

SICIL NOYA GORE INDEX

Ksa	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOGUM YERI	DOG.TAR .
13	3088999	ali beyaz	525 02 15	uskudar	izmit	09/09/30
12	3290599	hale gurler	172 15 18	kadikoy	bursa	09/08/32
3	3567999	nuri deniz	157 88 88	bakirkoy	giresun	12/07/35
1	4589999	iclal yildirim	160 02 74	aksaray	urfa	21/06/60
11	4800999	sabahat bilgin	588 55 66	yesilyurt	edirne	19/10/48
14	6123499	can kavuzlu	518 99 99	cerrahpasa	tokat	29/02/61
7	6139999	nimet altinel	589 05 66	topkapi	amasya	14/03/61
2	6678999	sevim boztas	589 09 88	karakoy	mersin	11/09/66
4	7188999	nurel dogan	157 88 99	capa	ankara	14/07/71
6	7569999	jale ozdemir	171 14 15	gayrettepe	artvin	15/02/75
9	7651999	salih celik	160 02 75	atakoy	kirikkale	15/02/76
5	8199799	kemal topuz	175 66 66	samatya	izmir	17/04/81
8	8896999	ozlem yildirim	589 61 06	findikzade	istanbul	12/10/88
10	8991099	kamil soydan	176 35 88	ayazaga	giresun	18/10/89

DEVAM<D>----MENU<M>

ISME GORE INDEX

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
13	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
14	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61
12	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
1	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
6	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
10	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
5	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
7	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61
4	nurel dogan	7188999	157 88 99	capa	ankara	14/07/71
3	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
8	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
11	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
9	salih celik	7651999	160 02 75	atakoy	kirikkale	15/02/76
2	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66

Şekil 10.10.Sicil No. ve adı soyadına göre İndeks

ADRESE GORE INDEX

Ksa	ADRES	SIC.NO	ADI SOYADI	TEL NO	DOGUM YERI	DOG.TAR.
1	aksaray	4589999	iclal yildirim	160 02 74	urfa	21/06/60
9	atakoy	7651999	salih celik	160 02 75	kirikkale	15/02/76
10	ayazaga	8991099	kamil soydan	176 35 88	giresun	18/10/89
3	bakirkoy	3567999	nuri deniz	157 88 88	giresun	12/07/35
4	capa	7188999	nurel dogan	157 88 99	ankara	14/07/71
14	cerrahpasa	6123499	can kavuzlu	518 99 99	tokat	29/02/61
8	findikzade	8896999	ozlem yildirim	589 61 06	istanbul	12/10/88
6	gayrettepe	7569999	jale ozdemir	171 14 15	artvin	15/02/75
12	kadikoy	3290599	hale gurler	172 15 18	bursa	09/08/32
2	karakoy	6678999	sevim boztas	589 09 88	mersin	11/09/66
5	samatya	8199799	kemal topuz	175 66 66	izmir	17/04/81
7	topkapi	6139999	nimet altinel	589 05 66	amasya	14/03/61
13	uskudar	3088999	ali beyaz	525 02 15	izmit	09/09/30
11	yesilyurt	4800999	sabahat bilgin	588 55 66	edirne	19/10/48

DEVAM<D>----MENU<M>

DOGUM TARIHINE GORE INDEX

Ksa	DOG.TAR.	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOGUM YERI
10	18/10/89	8991099	kamil soydan	176 35 88	ayazaga	giresun
8	12/10/88	8896999	ozlem yildirim	589 61 06	findikzade	istanbul
5	17/04/81	8199799	kemal topuz	175 66 66	samatya	izmir
9	15/02/76	7651999	salih celik	160 02 75	atakoy	kirikkale
6	15/02/75	7569999	jale ozdemir	171 14 15	gayrettepe	artvin
4	14/07/71	7188999	nurel dogan	157 88 99	capa	ankara
2	11/09/66	6678999	sevim boztas	589 09 88	karakoy	mersin
7	14/03/61	6139999	nimet altinel	589 05 66	topkapi	amasya
14	29/02/61	6123499	can kavuzlu	518 99 99	cerrahpasa	tokat
1	21/06/60	4589999	iclal yildirim	160 02 74	aksaray	urfa
11	19/10/48	4800999	sabahat bilgin	588 55 66	yesilyurt	edirne
3	12/07/35	3567999	nuri deniz	157 88 88	bakirkoy	giresun
12	09/08/32	3290599	hale gurler	172 15 18	kadikoy	bursa
13	09/09/30	3088999	ali beyaz	525 02 15	uskudar	izmit

Şekil 10.11. adres ve doğum tarihine göre indeksli kayıtlar.

TELEFON NOYA GORE INDEX

Ksa	TEL NO	SIC.NO	ADI SOYADI	ADRES	DOGUM YERI	DOG.TAR.
3	157 88 88	3567999	nuri deniz	bakirkoy	giresun	12/07/35
4	157 88 99	7188999	nurel dogan	capa	ankara	14/07/71
1	160 02 74	4589999	iclal yildirim	aksaray	urfa	21/06/60
9	160 02 75	7651999	salih celik	atakoy	kirikkale	15/02/76
6	171 14 15	7569999	jale ozdemir	gayrettepe	artvin	15/02/75
12	172 15 18	3290599	hale gurler	kadikoy	bursa	09/08/32
5	175 66 66	8199799	kemal topuz	samatya	izmir	17/04/81
10	176 35 88	8991099	kamil soydan	ayazaga	giresun	18/10/89
14	518 99 99	6123499	can kavuzlu	cerrahpasa	tokat	29/02/61
13	525 02 15	3088999	ali beyaz	uskudar	izmit	09/09/30
11	588 55 66	4800999	sabahat bilgin	yesilyurt	edirne	19/10/48
7	589 05 66	6139999	nimet altinel	topkapi	amasya	14/03/61
2	589 09 88	6678999	sevim boztas	karakoy	mersin	11/09/66
8	589 61 06	8896999	ozlem yildirim	findikzade	istanbul	12/10/88

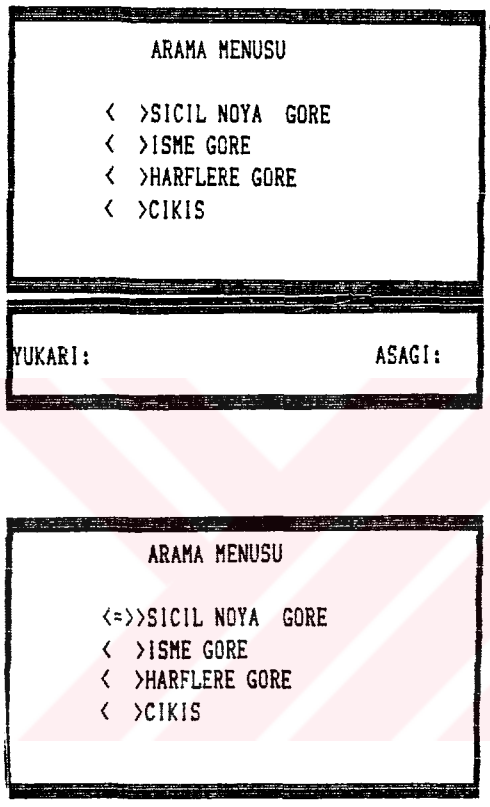
DEVAM<D>----MENU<M>

DOGUM YERINE GORE INDEX

Ksa	DOGUM YERI	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOG.TAR.
7	amasya	6139999	nimet altinel	589 05 66	topkapi	14/03/61
4	ankara	7188999	nurel dogan	157 88 99	capa	14/07/71
6	artvin	7569999	jale ozdemir	171 14 15	gayrettepe	15/02/75
12	bursa	3290599	hale gurler	172 15 18	kadikoy	09/08/32
11	edirne	4800999	sabahat bilgin	588 55 66	yesilyurt	19/10/48
10	giresun	8991099	kamil soydan	176 35 88	ayazaga	18/10/89
3	giresun	3567999	nuri deniz	157 88 88	bakirkoy	12/07/35
8	istanbul	8896999	ozlem yildirim	589 61 06	findikzade	12/10/88
5	izmir	8199799	kemal topuz	175 66 66	samatya	17/04/81
13	izmit	3088999	ali beyaz	525 02 15	uskudar	09/09/30
9	kirikkale	7651999	salih celik	160 02 75	atakoy	15/02/76
2	mersin	6678999	sevim boztas	589 09 88	karakoy	11/09/66
14	tokat	6123499	can kavuzlu	518 99 99	cerrahpasa	29/02/61
1	urfa	4589999	iclal yildirim	160 02 74	aksaray	21/06/60

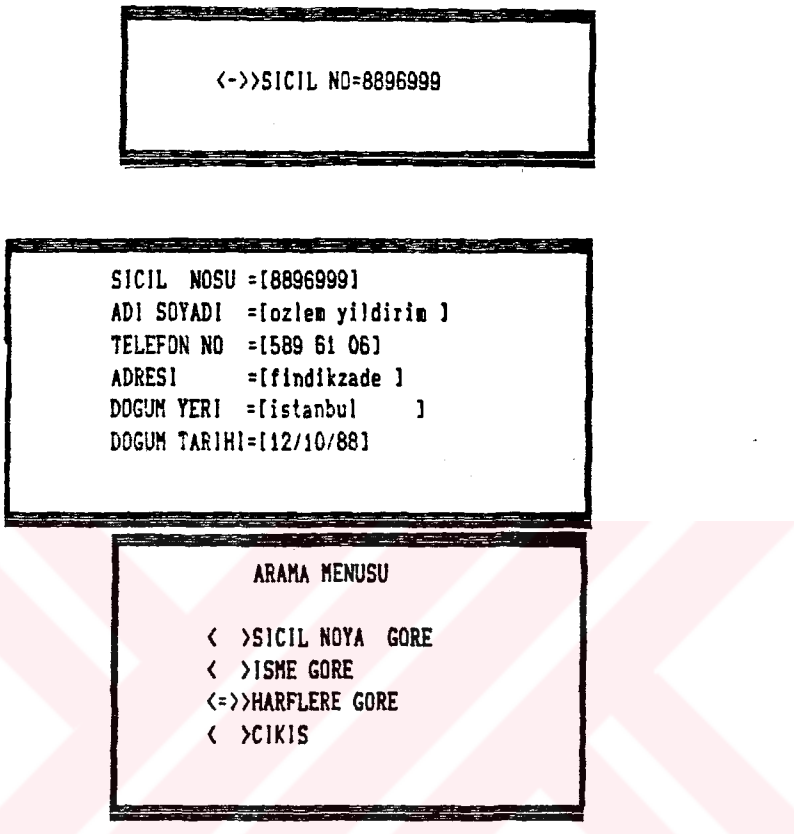
Şekil 10.12. Telefon numarası ve doğum yerine göre sıralanmış kayıtlar.

A: Ana menüden kayıt arama menüsüne geçilir.



Şekil 10.13.Kayıt arama menüsü

Arama menüsünde de görüldüğü gibi, yukarı↑ aşağı↓ tuşları kullanılarak arama menüsünden seçenek alınır. Sicil numarası, isim ve dosyadaki kayıtlarda bulunan bütün alanlarda sadece ilk harfleri yazarak, dosyada bulunan ve bu harflerle başlayan bütün alanlara erişilebilir.



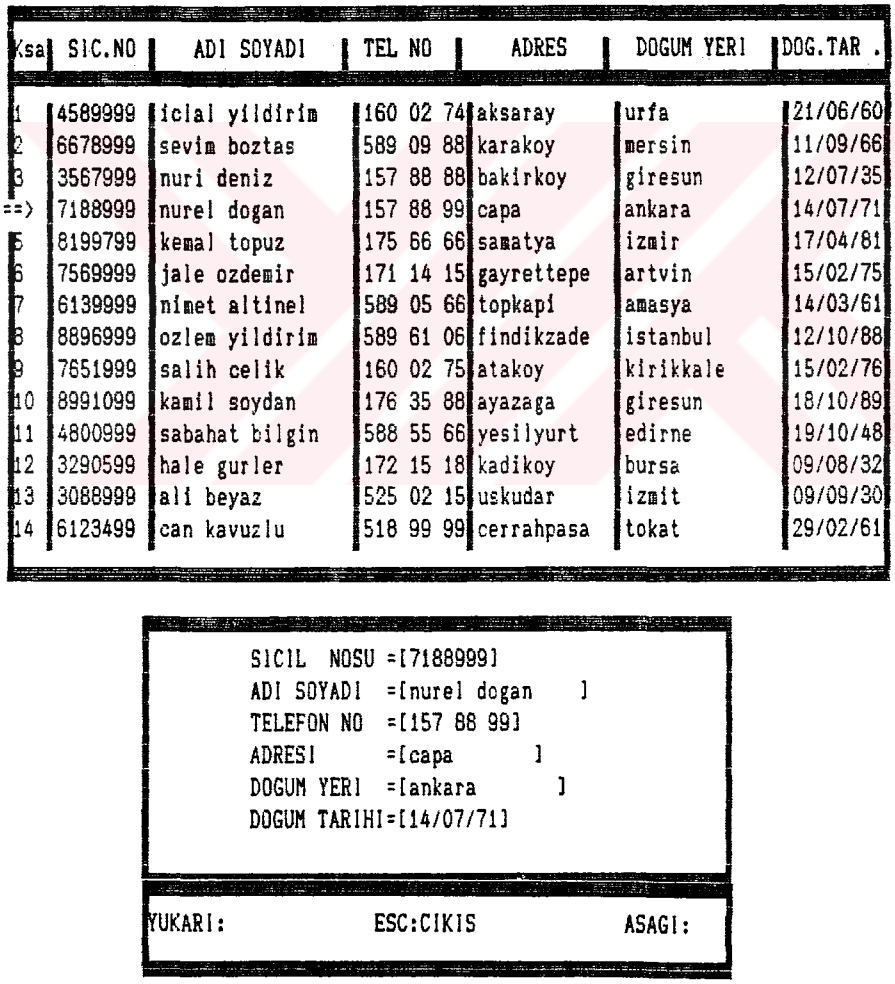
bas harfleri yaziniz=n

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
8	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61
9	nurel dogan	7188999	157 88 99	capa	ankara	14/07/71
10	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35

DEVAM<D>----MENU<M>

Şekil 10.14. Kayıt Arama Ekranları

E: E tuşuna basılarak ekranda kayıt değiştirme menüsüne geçilir. Kayıtlarla ilgili değişiklikler yukarı ↑ aşağı ↓ tuşları yardımıyla yapılır. Yukarı ↑ aşağı ↓ tuşlarıyla hareket edilerek istenilen alana ulaşıldığında RETURN tuşuna basılarak, o alandaki bilgi yeniden girilir. Bu menüden ESC tuşuna basılarak çıkılır.



Ksa	SIC.NO	ADI SOYADI	TEL NO	ADRES	DOGUM YERI	DOG.TAR .
1	4589999	iclal yildirim	160 02 74	aksaray	urfa	21/06/60
2	6678999	sevim boztas	589 09 88	karakoy	mersin	11/09/66
3	3567999	nuri deniz	157 88 88	bakirkoy	giresun	12/07/35
==>	7188999	nurel dogan	157 88 99	capa	ankara	14/07/71
5	8199799	kemal topuz	175 66 66	samatya	izmir	17/04/81
6	7569999	jale ozdemir	171 14 15	gayrettepe	artvin	15/02/75
7	6139999	nimet altinel	589 05 66	topkapi	amasya	14/03/61
8	8896999	ozlem yildirim	589 61 06	findikzade	istanbul	12/10/88
9	7651999	salih celik	160 02 75	atakoy	kirikkale	15/02/76
10	8991099	kamil soydan	176 35 88	ayazaga	giresun	18/10/89
11	4800999	sabahat bilgin	588 55 66	yesilyurt	edirne	19/10/48
12	3290599	hale gurler	172 15 18	kadikoy	bursa	09/08/32
13	3088999	ali beyaz	525 02 15	uskudar	izmit	09/09/30
14	6123499	can kavuzlu	518 99 99	cerrahpasa	tokat	29/02/61

Şekil 10.15. Kayıt değiştirme ekranı yukarı aşağı tuşları kullanılarak alanlar değiştirilir.

```
      SICIL  NOSU =[7188999]
      ADI SOYADI  =[nurel dogan    ]
      TELEFON NO  =[587 66 01]
      ADRESI      =[merter     ]
      DOGUM YERI  =[ankara       ]
      DOGUM TARIHI=[14/07/71]

YUKARI:            ESC:CIKIS            ASAGI:
```

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
2	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66
3	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
4	nurel dogan	7188999	587 66 01	merter	ankara	14/07/71
5	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
6	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
7	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61
8	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
9	salih celik	7651999	160 02 75	atakoy	kirikkale	15/02/76
10	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
11	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
12	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
13	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
14	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61

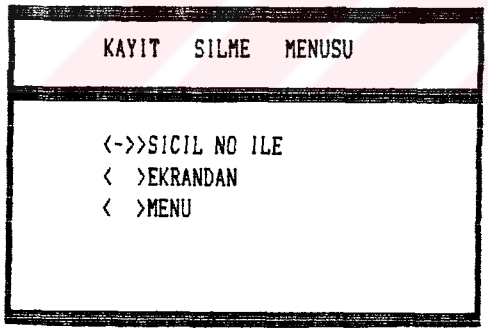
Ksa	TEL NO	SIC.NO	ADI SOYADI	ADRES	DOGUM YERI	DOG.TAR.
1	157 88 88	3567999	nuri deniz	bakirkoy	giresun	12/07/35
2	160 02 74	4589999	iclal yildirim	aksaray	urfa	21/06/60
3	160 02 75	7651999	salih celik	atakoy	kirikkale	15/02/76
4	171 14 15	7569999	jale ozdemir	gayrettepe	artvin	15/02/75
5	172 15 18	3290599	hale gurler	kadikoy	bursa	09/08/32
6	175 66 66	8199799	kemal topuz	samatya	izmir	17/04/81
7	176 35 88	8991099	kamil soydan	ayazaga	giresun	18/10/89
8	518 99 99	6123499	can kavuzlu	cerrahpasa	tokat	29/02/61
9	525 02 15	3088999	ali beyaz	uskudar	izmit	09/09/30
10	587 66 01	7188999	nurel dogan	merter	ankara	14/07/71
11	588 55 66	4800999	sabahat bilgin	yesilyurt	edirne	19/10/48
12	589 05 66	6139999	nimet altinel	topkapi	amasya	14/03/61
13	589 09 88	6678999	sevim boztas	karakoy	mersin	11/09/66
14	589 61 06	8896999	ozlem yildirim	findikzade	istanbul	12/10/88

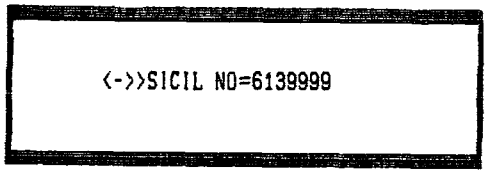
Şekil 10.16. Değiştirilen Kaydın Görüntülenmesi

S: Bu bölümde sicil numarasının hatırlanamayacağı düşünülerek, sicil numarası sorular silme olduğu gibi, ekrandan ok tuşları kullanılarak, istenilen kayıt alınıp dosyadan silme olanağı da vardır.

Ekrandan istediğimiz kaydı silerken, ekranda izlenmesi mümkündür. İstenilen kayıt seçildiğinde ekranda, o kaydın önünde ⟹ işareti görülür. Silinmesi istenen kayıt ekrandan silinerek, bütün kayıtlar ekranda görüntülenir.

Kayıt silme sicil numarasına göre yapıldığında, sicil numarası sorulur ve aynı sicil numaralı kayıt ekrana gelir. Dosyadan silinir.





Şekil 10.17. Sicil Numarası Alanı Sorularak Silme

```
SICIL  NOSU =[6139999]
ADI SOYADI  =[nimet altinel  ]
TELEFON NO  =[589 05 66]
ADRESI      =[topkapi    ]
DOGUM YERI  =[amasya       ]
DOGUM TARIHI=[14/03/61]
```

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
2	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66
3	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
4	nurel dogan	7188999	587 66 01	merter	ankara	14/07/71
5	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
6	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
7	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
8	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
9	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
10	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
11	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
12	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61

Şekil 10.18. Sicil numarası istenen kaydın ve ekrandan istenen kaydın silinmesi.

U: Ana dosyadan silinen kaydı geri almak için ana menüden Undel bölümü seçilerek silinen kayıtlar tekrar ele geçirilir. Dosyadaki kayıtlarının görüntülendiği ekrandan istenilen kayıt, yukarı ↑, aşağı ↓ tuşları kullanılarak alınır. Kayıt üzerinde ekranda ==> işareti görülür. kayıt dosyaya eklenir ana menüden G seçeneği alınarak, kayıt sayısının bir arttığı ve silinen kaydın tekrar dosyaya eklendiği görülür.

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	7651999	salih celik	160 02 75	atakoy	kirikkale	15/02/76
==>	6139999	nimet altinel	589 05 66	topkapi	amasya	14/03/61

GERI ALACAKMISINIZ?EVET<E>----HAYIR<H>

kayit sayisi=13

sayfa sayfa<S> tek tek <T>

Ksa	ADI SOYADI	SIC.NO	TEL NO	ADRES	DOGUM YERI	DOG.TAR.
1	iclal yildirim	4589999	160 02 74	aksaray	urfa	21/06/60
2	sevim boztas	6678999	589 09 88	karakoy	mersin	11/09/66
3	nuri deniz	3567999	157 88 88	bakirkoy	giresun	12/07/35
4	nurel dogan	7188999	587 66 01	merter	ankara	14/07/71
5	kemal topuz	8199799	175 66 66	samatya	izmir	17/04/81
6	jale ozdemir	7569999	171 14 15	gayrettepe	artvin	15/02/75
7	ozlem yildirim	8896999	589 61 06	findikzade	istanbul	12/10/88
8	kamil soydan	8991099	176 35 88	ayazaga	giresun	18/10/89
9	sabahat bilgin	4800999	588 55 66	yesilyurt	edirne	19/10/48
10	hale gurler	3290599	172 15 18	kadikoy	bursa	09/08/32
11	ali beyaz	3088999	525 02 15	uskudar	izmit	09/09/30
12	can kavuzlu	6123499	518 99 99	cerrahpasa	tokat	29/02/61
13	nimet altinel	6139999	589 05 66	topkapi	amasya	14/03/61

Şekil 10.19 Silinen kaydın tekrar dosyaya eklenerek görüntülenmesi.

SONUÇLAR VE ÖNERİLER

Hash adresleme metodu ile oluşturulan dosyaların tekrar düzenlenmesinden kaçınmak için geliştirilen metodlardan biri de lineer hash metodudur. Lineer Hash metodu, dosyaya kayıt eklense bile birincil alana yeni buketler ekleyerek sabit bir yükleme faktörü elde eder. Böylece yeni kayıtlar eklenirken, birincil alana yeni alanlarda eklenir. Kayıt sayısı ile birincil alanda kayıtlar için gerekli alan sayısının, oranı olan yükleme faktörü sabit kalır. Dosya büyürken sabit bir yükleme faktörü elde etmek, yenilik getiren bir fikirdir.

Genel hash metodunda, hash tollosu sık sık, organize edilir. Lineer hash metodunda ise kayıtlar buketlere yerleştirilirken, tablo yalnız bir kez oluşturulur. Zaman açısından çok etkin bir metoddur.

Lineer hash metodu Richard metodu ile kıyaslandığında Richard metodunda, yükleme faktörü $\alpha=m/n+1$ dir. m anahtar sayısı, n ise tablo uzunluğudur. Bu metodda yükleme faktörü n'e bağlı olarak değişir. Lineer hash metodunda yükleme faktörü kayıt sayısına bağlı olmadığı gibi bir kaydı aramak için, gerekli araştırma sayısı da kayıt sayısına bağlı değildir. Örneğin: Bkfr= 50, yükleme faktörü %75 ise, dosyada olan bir kaydı aramak için araştırma sayısı 1.05 dir. Olmayan kaydı aramak içinse arama sayısı 1.27 dir. Bu arama, hiçbir zaman kayıt sayısına bağlı değildir. Kayıt sayısı ne olursa olsun aynı performans alınır. Richard metodunda, T tablosunda ki $T(h_s)$ anahtarını bulmak için s+1 inceleme yapılır. Lineer hash metodunda ise hash sayısının son, k yada k+1 dijiti alanırak, bukete direk olarak erişilir. Erişme hızının ya-

yaşlaması, hash sayılarının düzenli olarak dağılmamasından ileri gelir. k dijitle etiketlenen buket sayısı, k+1 dijitle etiketlenen buket sayısından daha fazladır.

Hash fonksiyonları bir dosyayı sıralamak için kullanılmazlar. Sıralama işlemleri ve aralık aramalarında B^+ ağaçları daha etkin bir metoddur. Anahtar alanlara göre sıralanmış bir liste varsa dosyayı okuma zamanı

Tx=nx 1.05 x (s+r+dtt) dir.

gerçek veri hiçbir zaman düzenli değildir.

KAYNAKLAR

[1] SALLAM, H.B and PETERS J.F. Compleat C, 1986

[2] Herbert Schilds, Cmade Easy 1985

[3] Herbert Schildt, Advanced Turbo C, 1987

[4] Turbo C Reference guide, Version 2.0, 1988

Ek- A Veritab Program Listesi

```
  #include<string.h>
#include<stdio.h>
#include<fcntl.h>
#include<sys\stat.h>
#include<sys\types.h>
#include<alloc.h>
#include<stdlib.h>
#include<io.h>
#include<conio.h>
#include<graphics.h>
#include<ctype.h>
#define WINDOW011() window(30,13,40,14);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW0011() window(50,13,60,14);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW01() window(1,23,80,24);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWA0() window(1,5,80,24);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWD() window(24,15,42,16);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWE() window(24,16,42,17);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWF() window(24,17,42,18);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWG() window(24,18,42,19);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWH() window(24,19,42,20);textbackground(BLACK);textcolor(YELLOW)
#define WINDOWI() window(24,20,42,21);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW1() window(1,1,80,2);textbackground(RED);textcolor(YELLOW)
#define WINDOW2() window(1,3,80,25);textbackground(BLACK);textcolor(GREEN)
#define WINDOW002() window(1,1,80,25);textbackground(BLACK);textcolor(GREEN)
#define WINDOW4() window(28,7,32,8);textbackground(BLACK);textcolor(GREEN)
#define WINDOW04() window(32,7,33,8);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW3() window(28,6,29,7);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW12() window(29,6,41,7);textbackground(BLACK);textcolor(GREEN)
#define WINDOW126() window(29,6,43,7);textbackground(BLACK);textcolor(GREEN)
#define WINDOW127() window(1,4,68,19);textbackground(BLACK);textcolor(GREEN)
#define WINDOW5() window(28,8,29,9);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW13() window(29,8,34,9);textbackground(BLACK);textcolor(GREEN)
#define WINDOW6() window(28,9,29,10);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW14() window(29,9,35,10);textbackground(BLACK);textcolor(GREEN)
#define WINDOW7() window(28,10,29,11);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW8() window(29,10,33,11);textbackground(BLACK);textcolor(GREEN)
#define WINDOW9() window(28,11,29,12);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW10() window(29,11,33,12);textbackground(BLACK);textcolor(GREEN)
#define WINDOW11() window(28,12,29,13);textbackground(BLACK);textcolor(GREEN)
#define WINDOW15() window(29,12,30,13);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW16() window(30,12,36,13);textbackground(BLACK);textcolor(GREEN)
#define WINDOW17() window(28,13,29,14);textbackground(BLACK);textcolor(GREEN)
#define WINDOW18() window(29,13,30,14);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW19() window(30,13,38,14);textbackground(BLACK);textcolor(GREEN)
```

```
#define WINDOW23() window(29,14,35,15);textbackground(BLACK);textcolor(GREEN)
#define WINDOW22() window(28,14,29,15);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW24() window(3,22,17,23);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW224() window(1,21,68,24);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW241() window(17,22,18,23);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW240() window(2,24,80,25);textbackground(BLACK);textcolor(GREEN)
#define WINDOW024() window(1,21,11,22);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW25() window(24,12,56,14);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW26() window(1,24,20,25);textbackground(BLACK);textcolor(YELLOW)
#define WINDOW02() window(1,1,12,2);textbackground(BLACK);textcolor(YELLOW)
 typedef struct kayit_ismi{
        int ksay;
        char sicno[8];
        char isim[17];
        char telno[10];
        char cad[12];
        char dyer[14];
        char dtar[9];
        }kayit;
 struct bucket{
        int k1;
        char sicno1[8];
        struct bucket *link;
        };
 struct ket{
        int k3;
        char is3[17];
        struct ket *ink1;
        };
  struct ket4{
        int k4;
        char te14[10];
        struct ket4 *ink4;
        };
  struct ket5{
        int k5;
        char ca5[12];
        struct ket5 *ink5;
        };
```

```
typedef struct pt{
        int k2;
        char sicno2[8];
        }yit;
        typedef struct qm
        {
        char sio[8];
        }sal;

typedef struct
        {
        int in;
        char isim[17];
        }is;
typedef struct
        {
        int tn;
        char telno[10];
        }no;

typedef struct
        {
        int dr;
        char cad[12];
        }ad;
typedef struct
        {
        int dy;
        char dyer[14];
        }yer;
typedef struct
        {
        int tar;
        char dtar[9];
        }tarih;
 struct ket6{
        int k6;
        char dy6[14];
        struct ket6 *ink6;
        };
  struct ket7{
        int k7;
        char dt7[9];
        struct ket7 *ink7;
        };
```

```
typedef struct
        {
        int r;
        }yi;
typedef struct
        {
        int sil;
        }yil;
typedef struct
        {
        int tnt;
        char onlet[10];
        }not;
typedef struct ffblk {
char ff_reserved[21];
char ff_attrib;
int ff_ftime;
int ff_fdate;
long ff_fsize;
char ff_name[13];
}/*ffblk;*/;
#define AL1 malloc(sizeof(struct bucket))
#define ks 25
kayit *ku;
struct bucket *x[ks],*head[ks];
struct ket *x1[ks],*ead1[ks];
struct ket4 *x4[ks],*ead4[ks];
struct ket5 *x5[ks],*ead5[ks];
struct ket6 *x6[ks],*ead6[ks];
struct ket7 *x7[ks],*ead7[ks];
struct bucket *ekle();
struct ket *t1();
struct ket4 *t4();
struct ket5 *t5();
struct ket6 *t6();
struct ket7 *t7();
struct bucket *new();
struct bucket *del();
struct bucket *ara();
struct ket *bul();
kayit *doku;
kayit *rs;
kayit *srs;
yit *het[ks];
sal *opp;
```

```
int iet,zet,kno,bh,jit,y=16,h=0,s,k,ma,vop;
int a,bkfr=3,st;
float lf=0.6666667;
char rt[15];
char *bs;
char *si,*dat,*dayer,*iasim,*tan,*caad;
sal *aw;
yi *g;
kayit *cade,*yerdi,*dex,*giris,*cikis;
kayit *sira,*doer,*doeer,*tarihi;
kayit *reno,*tade;
kayit *degis,*ism,*harf;
int oya,ur=1,zet,lk=179,aij,sa,s,k,ff,vf,sakla;
char *kya,*ces,*kra,*yy,sec,*kisec,ksec;
is *yensir[ks],*siram[ks],*yem,*inim[ks];
no *tf[ks];
ad *res[ks];
yer *dyeri[ks];
tarih *trh[ks];
yi *g;
yii *gi;
not *ahia[ks];
main()
{
int bn;
char ew;
clrscr();
s=dosoku();
st=lf*bkfr;
if(s!=0){clrscr();
printf("\nLINEER HASH METODUNU TAKIP ETMEK ISTIYORMUSUNUZ\n");
ew=getch();
if(ew=='e')
{
s=dosoku();
st=lf*bkfr;
printf("load faktoru=%f  buket faktoru=%d",lf,bkfr);
ma=s/st;
printf("kayit-sayisi=%d  buket sayisi=%d",s,ma);getch();
jit=hesap(ma);
printf("dijit sayisi=%d",jit);
bn=us(2,jit);
bh=ma-bn;
printf("\nsinir deger=%d",bh);
indoku();
ilk();
yaz();
}
```

```
indoku();ilk();
}
else  printf("kayit yapilmamis");
menu();
}
kayit_girisi()
{
char mh,dw,*d,*hia,*nb,*wnb,*wnb6,*oi,*shia;
int handle1,handle2,handle3,handle5,handle6,handle7;
int handle8,handle9;
int o,ali,ii=0;
FILE *stream1,*stream2,*stream3,*stream5,*stream6,*stream7;
FILE *stream8,*stream9;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
stream2=fopen("sayi","r+b");
handle2=fileno(stream2);
g=(yi *)malloc(sizeof(yi));
WINDOW1();clrscr();
if (stream2==NULL){/* hatali*/printf("DOSYA YARATMADINIZ \n");
printf("DOSYA YARATMADAN KAYIT YAPILAMAZ\n");getch();menu();}
stream3=fopen("index","r+b");
handle3=fileno(stream3);
stream5=fopen("telno","r+b");
handle5=fileno(stream5);
stream6=fopen("dogu","r+b");
handle6=fileno(stream6);
stream7=fopen("adres","r+b");
handle7=fileno(stream7);
stream8=fopen("dogtar","r+b");
handle8=fileno(stream8);
stream9=fopen("sno","r+b");
handle9=fileno(stream9);
WINDOW002();
clrscr();
do{
giris=(kayit *)malloc(sizeof(kayit));
WINDOW002();
printf("\n\n\n\n\n\n\n\n\n");
```

```
printf("                                                    \n");
printf("        KAYIT-SAYISI=                               \n");
printf("                                                    \n");
printf("        KAYIT-GIRISI MENUSU                         \n");
printf("                                                    \n");
printf("                                                    \n");
printf("         SICIL  NOSU =[       ]                     \n");
printf("         ADI SOYADI  =[              ]              \n");
printf("         TELEFON NO  =[         ]                   \n");
printf("         ADRESI      =[           ]                 \n");
printf("         DOGUM YERI  =[             ]               \n");
printf("         DOGUM TARIHI=[        ]                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
```

```
WINDOW002();
gotoxy(25,10);printf("%d",s);
WINDOWD();oya=1;input(giris->sicno,8);
WINDOWE();oya=2;input(giris->isim,17);
WINDOW002();
printf("TEL NOYU (XXX XX XX) FORMATINDA YAZIN");delay(1);
printf("                                      ");
WINDOWF();
oya=3;
input(giris->telno,10);
WINDOWG();
oya=4;
input(giris->cad,12);
WINDOWH();
oya=5;
input(giris->dyer,14);
WINDOW002();
gotxy(1,1);printf("DOGUM TARIHINI (XX/XX/XX) FORMATINDA YAZIN");sleep(1);
gotoxy(1,1);printf("                                                ");
WINDOWI();
oya=6;
input(giris->dtar,9);
s++;k=s-1;
trh[k]=(tarih*)malloc(sizeof(tarih));
strcpy(trh[k]->dtar,giris->dtar);
trh[k]->tar=s;
d=giris->dtar;
nb=strtok(d,"///");
```

```
while(nb!=NULL)
{
ii++;
switch(ii)
{
case 1:{wnb=nb;}
break;
case 2:{wnb6=nb;

}break;
case 3:{ o1=nb;
}break;
}
nb=strtok(NULL,"///");
}
ii=0;
shia=strcat(o1,wnb6);hia=strcat(shia,wnb);
ahia[k]=(not *)malloc(sizeof(not));
strcpy(ahia[k]->onlet,hia);
ahia[k]->tnt=s;
siram[k]=(is *)malloc(sizeof(is));
siram[k]->in=s;
strcpy(siram[k]->isim,giris->isim);
het[k]=(yit *)malloc(sizeof(yit));
het[k]->k2=s;
strcpy(het[k]->sicno2,giris->sicno);
res[k]=(ad *)malloc(sizeof(ad));
res[k]->dr=s;
strcpy(res[k]->cad,giris->cad);
dyeri[k]=(yer *)malloc(sizeof(yer));
dyeri[k]->dy=s;
strcpy(dyeri[k]->dyer,giris->dyer);
tf[k]=(no *)malloc(sizeof(no));
tf[k]->tn=s;
strcpy(tf[k]->telno,giris->telno);
g->r=s;
giris->ksay=s;
strcpy(giris->dtar,trh[k]->dtar);
strcpy(giris->isim,siram[k]->isim);
lseek(handle1,(s-1)*sizeof(kayit),0);
write(handle1,giris,sizeof(kayit));
WINDOW002();gotoxy(3,23);printf("TEKRAR KAYIT :<T>    MENU<M>");
mh=getch();
}while(mh=='t' || mh=='T');
g->r=s;
```

```
fseek(stream2,0L,0);
write(handle2,g,sizeof(yi));
if(s!=1)
{
 sort(siram,0,s-1); snt(het,0,s-1);
sirt(tf,0,s-1);s4(ahia,0,s-1);dog(dyeri,0,s-1);nor(res,0,s-1);
}
if(s==1){fseek(stream3,0L,2);write(handle3,siram[k],sizeof(is));
fseek(stream5,0L,2);write(handle5,tf[k],sizeof(no));
fseek(stream6,0L,2);write(handle6,dyeri[k],sizeof(yer));
fseek(stream8,0L,2);write(handle8,ahia[k],sizeof(not));
fseek(stream9,0L,2);write(handle9,het[k],sizeof(yit));
fseek(stream7,0L,2);write(handle7,res[k],sizeof(ad));
}
fclose(stream1);
fclose(stream2);
fclose(stream3);
fclose(stream5);
fclose(stream6);
fclose(stream7);
fclose(stream8);
fclose(stream9);
ilk();
menu();
}
input(gi,uzun)
char *gi;
int uzun;
{
char p[50];
gets(p);
if(strlen(p) >=uzun)
{
WINDOW02();cprintf("GIRIS HATALI...");delay(2);
WINDOW02();clrscr();
WINDOW02();cprintf("GIRIS HATALI...");delay(2);
WINDOW02();clrscr();
WINDOW02();cprintf("GIRIS HATALI...");delay(2);
WINDOW02();clrscr();
do{
switch(oya)
{
case 1:
{
WINDOW0();
```

```
clrscr();
cprintf("[       ]\n");
WINDOWD();
gets(p);
}
break;
case 2:
{
WINDOWE();
clrscr();
printf("[              ]\n");
WINDOWE();
gets(p);
}
break;
case 3:
{
WINDOWF();
clrscr();
cprintf("[         ]\n");
WINDOWF();
gets(p);
}
break;
case 4:
{
WINDOWG();
clrscr();
cprintf("[          ]\n");
WINDOWG();
gets(p);
}
break;
case 5:
{
WINDOWH();
clrscr();
cprintf("[            ]\n");
WINDOWH();
gets(p);
}
break;
case 6:
{
WINDOWI();
```

```
gets(p);
}
break;
}
}while(strlen(p)>=uzun);
}
strcpy(gi,p);
}

undel()
{
char *d,*hia,*nt,*wnt,*wnb6,*ct,*shia;
int ii=0;
int kns,ji;
int kii,nur,anur,handle0,handle1,handle2,handle4;
char ac,ev;
FILE *stream0,*stream1,*stream2,*stream4;
kayit *il;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
stream2=fopen("sayi","r+b");
handle2=fileno(stream2);
stream4=fopen("silen","r+b");
handle4=fileno(stream4);
stream0=fopen("si","r+b");
handle0=fileno(stream0);
WINDOW002();
clrscr();
WINDOW1();
clrscr();
il=(kayit *)malloc(sizeof(kayit));
ekryaz();
gi=(yil *)malloc(sizeof(yil *));
fseek(stream0,0L,0);
read(handle0,gi,sizeof(yil));
sa=gi->sil;
for(nur=1;nur<=sa;++nur)
{
lseek(handle4,(nur-1)*sizeof(kayit),0);
read(handle4,il,sizeof(kayit));
printf("%-3d%-8s%-17s%-9s%-12s%-14s%-8s\n",nur,il->sicno,il->isim,il->telno,il->ead,il->dyer,il->otar);
}
printf("                                                                    \n");
```

```
anur=qyer(1,3);gotoxy(1,anur);printf("  >");
sleep(1);delline();
gotoxy(1,25);printf("\tGERI ALACAKMISINIZ?EVET(\E)----HAYIR(\H)");
ev=getch();
if(ev=='e' || ev=='E')
{
s=s+1;sa=sa-1;
lseek(handle4,(anur-1)*sizeof(kayit),0);
read(handle4,il,sizeof(kayit));
lseek(handle1,(s-1)*sizeof(kayit),0);
write(handle1,il,sizeof(kayit));
if(sa>1)
{
for(kil=anur+1;kil<=s;++kil)
{
lseek(handle4,(kil-1)*sizeof(kayit),0);
read(handle4,il,sizeof(kayit));
lseek(handle4,(kil-2)*sizeof(kayit),0);
write(handle4,il,sizeof(kayit));
}
}
g->r=s;
fseek(stream2,0L,0);
write(handle2,g,sizeof(yi));
gl->sil=sa;
fseek(stream0,0L,0);
write(handle0,gl,sizeof(yil));
k=s-1;
trh[k]=(tarih*)malloc(sizeof(tarih));
strcpy(trh[k]->dtar,il->dtar);
trh[k]->tar=s;
d=il->dtar;
nb=strtok(d,"///");
while(nb!=NULL)
{ii++;
switch(ii)
{
case 1:{wnb=nb;}
break;
case 2:{wnb6=nb;
}break;
case 3:{ ol=nb;
}break;
}
nb=strtok(NULL,"///");
}
```

```
ii=0;
shia=strcat(o1,wnb6);hia=strcat(shia,wnb);
ahia[k]=(not *)malloc(sizeof(not));
strcpy(ahia[k]->onlet,hia);
ahia[k]->tnt=s;
siram[k]=(is *)malloc(sizeof(is));
siram[k]->in=s;
strcpy(siram[k]->isim,il->isim);
        het[k]=(yit *)malloc(sizeof(yit));
het[k]->k2=s;
strcpy(het[k]->sicno2,il->sicno);
res[k]=(ad *)malloc(sizeof(ad));
res[k]->dr=s;
strcpy(res[k]->cad,il->cad);
dyeri[k]=(yer *)malloc(sizeof(yer));
dyeri[k]->dy=s;
strcpy(dyeri[k]->dyer,il->dyer);
tf[k]=(no *)malloc(sizeof(no));
tf[k]->tn=s;
strcpy(tf[k]->telno,il->telno);
strcpy(il->dtar,trh[k]->dtar);
strcpy(il->isim,siram[k]->isim);
 sort(siram,0,s-1); snt(het,0,s-1);
sirt(tf,0,s-1);s4(ahia,0,s-1);dog(dyeri,0,s-1);nor(res,0,s-1);
indoku();
ilk();
}
fclose(stream4);
fclose(stream2);
fclose(stream1);
menu();
}



kayit_okuma()
{
 int ti,j,handle1,l;
char bv;
kayit *roku;
FILE *stream1;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
WINDOW002();clrscr();
printf("kayit sayisi=%d",s);
printf("\n\n\t\tsayfa sayfa<S> tek tek <T>\n\n\n");
```

```
bv=getche();if(bv=='s' || bv=='S') {
WINDOW002();clrscr();
ekryaz();
for(ti=0;ti<s;++ti)
{roku=(kayit *)malloc(sizeof(kayit));
read(handle1,roku,sizeof(kayit));
printf("|%-3d|%-16s|%-8s|%-9s|%-12s|%-14s|%-9s|\n",ti+1,roku->isim,roku->sicno,roku->telno
}
printf("└──────────────────────────────────────────────────────────────────────────────┘\n");
getch();
}
else{ WINDOW002();clrscr();
do{ for(ti=0;ti<s;++ti)
{rs=(kayit *)malloc(sizeof(kayit));
read(handle1,rs,sizeof(kayit));
printf("\n\n\t\t%d inci kayit",ti+1);
baslik();fas(rs);
getch();
if(ti==s-1)
{printf("kayitlar bu kadar");
sleep(2);menu();}
}
printf("\n\n\tdevam(d) menu(m)");bv=getch();
}while(bv=='d' || bv=='D');
}
fclose(stream1);

eż()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────────┐\n");
printf("|Ksa| TEL NO | SIC.NO |     ADI SOYADI     |      ADRES      |  DOGUM YERI  | DOG.TAR. |\n");
printf("└──────────────────────────────────────────────────────────────────────────────┘\n");
       WINDOWA0();
}
xz()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────────┐\n");
printf("|Ksa|     ADRES      | SIC.NO |     ADI SOYADI     | TEL NO  |  DOGUM YERI  | DOG.TAR. |\n");
printf("└──────────────────────────────────────────────────────────────────────────────┘\n");
       WINDOWA0();
}
xz1()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────────┐\n");
printf("|Ksa|  DOGUM YERI  | SIC.NO |     ADI SOYADI     | TEL NO  |      ADRES      | DOG.TAR. |\n");
printf("└──────────────────────────────────────────────────────────────────────────────┘\n");
WINDOWA0();
}
```

```
xz2()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────┐\n");
printf("  Ksa│ DOG.TAR. │ SIC.NO │      ADI SOYADI      │ TEL NO │      ADRES      │ DOGUM YERI │\n");
printf("  └──────────────────────────────────────────────────────────────────────────┘\n");
        WINDOWAO();
}
ekryaz()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────┐\n");
printf("  Ksa│      ADI SOYADI      │ SIC.NO │ TEL NO │      ADRES      │ DOGUM YERI │ DOG.TAR. │\n");
printf("  └──────────────────────────────────────────────────────────────────────────┘\n");
        WINDOWAO();
}
xz3()
{
printf("\n┌──────────────────────────────────────────────────────────────────────────┐\n");
printf("  Ksa│ SIC.NO │      ADI SOYADI      │ TEL NO │      ADRES      │ DOGUM YERI │ DOG.TAR . │\n");
printf("  └──────────────────────────────────────────────────────────────────────────┘\n");
        WINDOWAO();
}

 siralama()
 {
  int ji,handle1,eddd,ddd,edyq,dyq,ebe,auq,ee,uq,jh;
int dt1,ted;
int se,suq,sec;
FILE *stream1;
do{
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
  WINDOW002();clrscr();printf("\n\n\n\n\n\n\n");
printf("\t\t┌──────────────────────────────┐\n");
printf("\t\t│      SIRALAMA MENUSU         │\n");
ama();
sec=qyer(27,22);gotoxy(27,sec);printf("->");sleep(1);
switch(sec)
{
case 12:{
WINDOW002();clrscr();
ekryaz();
rs=(kayit *)malloc(sizeof(kayit));
for(uq=0;uq<s;++uq)
{
```

```
 (uq=0;uq<s;++uq)
{
ee=siram[uq]->in;
lseek(handle1,(ee-1)*sizeof(kayit),0);read(handle1,rs,sizeof(kayit));
printf("|%-3d|%-16s|%-8s|%-9s|%-12s|%-14s|%-9s|\n",uq+1,rs->isim,rs->sicno,rs->telno,rs->cad,rs->dye
}
printf("|-----------------------------------------------------------------------------|\n");
WINDOW002();
fclose(stream1);}
break;
case 11:{
WINDOW002();clrscr();
xz3();
srs=(kayit *)malloc(sizeof(kayit));
for(suq=0;suq<s;++suq)
{
se=het[suq]->k2;
lseek(handle1,(se-1)*sizeof(kayit),0);read(handle1,srs,sizeof(kayit));
printf("|%-3d|%-8s|%-16s|%-9s|%-12s|%-14s|%-9s|\n",suq+1,srs->sicno,srs->isim,
srs->telno,srs->cad,srs->dyer,srs->dtar);
}
printf("|-----------------------------------------------------------------------------|\n");
WINDOW002();
fclose(stream1);}
break;
case 13:
{
WINDOW002();
clrscr();
ez();
sira=(kayit *)malloc(sizeof(kayit));
for(auq=0;auq<s;++auq)
{
ebe=tf[auq]->tn;
lseek(handle1,(ebe-1)*sizeof(kayit),0);read(handle1,sira,sizeof(kayit));
printf("|%-3d|%-9s|%-8s|%-16s|%-12s|%-14s|%-9s|\n",(auq+1),sira->telno,sira->sicno,sira->isim,
sira->cad,sira->dyer,sira->dtar);
}
printf("|-----------------------------------------------------------------------------|\n");
WINDOW002();
fclose(stream1);
}
break;
```

```
case 15:
{
WINDOW002();
clrscr();
xz1();
doer=(kayit *)malloc(sizeof(kayit));
for(dyq=0;dyq<s;++dyq)
{
ddd=dyer1[dyq]->dy;
lseek(handle1,(ddd-1)*sizeof(kayit),0);read(handle1,doer,sizeof(kayit));
printf("|%-3d|%-14s|%-8s|%-17s|%-9s|%-12s|%-8s|\n",(dyq+1),doer->dyer,doer->sicno,doer->isim,
doer->telno,doer->cad,doer->dtar);
}
printf("|--------------------------------------------------------------------------------|\n");
WINDOW002();
fclose(stream1);
}
break;
case 14:
{
WINDOW002();
clrscr();
xz();
doeer=(kayit *)malloc(sizeof(kayit));
for(edyq=0;edyq<s;++edyq)
{
eddd=res[edyq]->dr;
lseek(handle1,(eddd-1)*sizeof(kayit),0);read(handle1,doeer,sizeof(kayit));
printf("|%-3d|%-12s|%-8s|%-17s|%-9s|%-14s|%-8s|\n",(edyq+1),doeer->cad,doeer->sicno,doeer->
isim,doeer->telno,doeer->dyer,doeer->dtar);
}
printf("|--------------------------------------------------------------------------------|\n");
WINDOW002();
fclose(stream1);
}
break;
case 16:
{
WINDOW002();
clrscr();
```

```
xz2();
tarihi=(kayit *)malloc(sizeof(kayit));
for(ted=0;ted<s;++ted)
{
dti=ahia[ted]->tnt;
lseek(handle1,(dti-1)*sizeof(kayit),0);read(handle1,tarihi,sizeof(kayit));
printf("|%-3d|%-8s|%-8s|%-17s|%-9s|%-12s|%-14s|\n",(ted+1),tarihi->dtar,tarihi->sicno,tarihi
->isim,tarihi->telno,tarihi->cad,tarihi->dyer);
}
printf("|                                                                                |\n");
WINDOW002();
fclose(stream1);
}
break;
case 17:
{
WINDOW024();sleep(1);
printf("\n\t\tMENUYE DONUYORUM");
}
break;
}
WINDOW002();
gotoxy(1,22);printf("\n\n\tDEVAM<D>----MENU<M>");jh=getch();
}while(jh=='d' || jh=='D');
}
cyer(q2,q1)
int q1,q2;
{
int c=0;
char *s0=" ";
gotoxy(q2,q1);
while(c!=13)
{
*s0=getch();
c=*s0;
if(c==72){
gotoxy(q2,q1-1);q1=q1-1;
}
if(c==80){
gotoxy(q2,q1+1);q1=q1+1;
}
}
return(q1);
}
```

```
yaz()
{
int fi=0;
while(fi<ma)
{
printf("\nbuketteki adresi=%d\n",fi);
printf("_____\n");
di(x[fi]);
fi++;getch();
}
}
di(op)
struct bucket *op;
{
printf("\n");
while(op) {
printf("**sicil nosu=%s **sirasi=%d\n",op->sicnoi,op->ki);
op=op->link;
}
}

yaz4()
{
int fi=0;
WINDOW002();clrscr();
while(fi<ma)
{
printf("buketteki adresi=%d",fi);
di4(x4[fi]);
fi++;getch();
}
}
di4(op4)
struct ket4 *op4;
{
while(op4) {
printf("\n telno=%s disk adresi=%d",op4->tel4,op4->k4);
op4=op4->ink4;
}
WINDOW002();
}
yaz5()
{
int fi=0;
WINDOW002();clrscr();
```

```
xz();
while(fi<ma)
{
x5[fi]=ead5[fi];
di5(x5[fi]);
fi++;getch();
}
}
di5(op5)
struct ket5 *op5;
{
int k15;
int handle1;
FILE *stream1;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
while(op5!=NULL) {
k15=op5->k5;
lseek(handle1,(k15-1)*sizeof(kayit),0);
read(handle1,srs,sizeof(kayit));
printf(" %-3d|%-12s|%-8s|%-17s|%-9s|%-14s|%-8s|\n",doeer->ksay,doeer->cad,doeer->sicno,doeer
->isim,doeer->telno,doeer->dyer,doeer->dtar);
op5=op5->ink5;
}
WINDOW002();
}
yaz6()
{
int fi=0;
WINDOW002();clrscr();
xz1();
while(fi<ma)
{
x6[fi]=ead6[fi];
di6(x6[fi]);
fi++;getch();
}
}
di6(op6)
struct ket6 *op6;
{
int k16;
int handle1;
FILE *stream1;
```

```
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
while(op6!=NULL) {
k16=op6->k6;
lseek(handle1,(k16-1)*sizeof(kayit),0);
read(handle1,srs,sizeof(kayit));
printf(" %-3d|%14s|%-8s|%-17s|%-9s|%-12s|%-8s|\n",doer->ksay,doer->dyer,doer->sicno,doer->is
im,doer->telno,doer->cad,doer->dtar);
op6=op6->ink6;
}
WINDOW002();
}
yaz7()
{
int fi=0;
WINDOW002();clrscr();
xz2();
while(fi<ma)
{
x7[fi]=ead7[fi];
di7(x7[fi]);
fi++;getch();
}
}
di7(op7)
struct ket7 *op7;
{
int k17;
int handle1;
FILE *stream1;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
while(op7!=NULL) {
k17=op7->k7;
lseek(handle1,(k17-1)*sizeof(kayit),0);
read(handle1,tarihi,sizeof(kayit));
printf(" %-3d|%-8s|%-8s|%-17s|%-9s|%-12s|%-14s|\n",k17,tarihi->dtar,tarihi->sicno,tarihi->is
im,tarihi->telno,tarihi->cad,tarihi->dyer);
op7=op7->ink7;
}
WINDOW002();
getch();
}
```

```
kayit_silme()
{
int to=0,lor=0,kns,opl,tt,huq,ww,hj,ii,ji,q,b,yy,pn,ki,di,er;
int handle0,handle1,handle2,handle3,handle4,handle7;
int llt;
int a,h,ban,xs,wj;
int sw,f,gh;
char ev,ac,ssili//j;
yil *gl;
FILE *stream0,*stream1,*stream2,*stream3,*stream4,*stream7;
struct bucket *lk,*son;
kayit *ikiu;
kayit *ri9;
kayit *ri10;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
stream2=fopen("sayi","r+b");
handle2=fileno(stream2);
stream0=fopen("si","r+b");
handle0=fileno(stream0);
stream4=fopen("silen","r+b");
handle4=fileno(stream4);
ri10=(kayit *)malloc(sizeof(kayit));
s=dosoku();
sa=sioku();
WINDOW002();
clrscr();
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                                                    \n");
sw=qyer(28,21);gotoxy(28,sw);printf("->");sleep(1);
switch(sw)
{
case 18:
{
ur=1;
aw=(sal *)malloc(sizeof(sal));
s=dosoku();
st=lf*bkfr;
ma=s/st;
```

```
      KAYIT   SILME   MENUSU

      <  >SICIL NO ILE
      <  >EKRANDAN
      <  >MENU
```

```
jit=hesap(ma);
ban=us(2,jit);
bh=ma-ban;
WINDOW002();
clrscr();
xz3();
ikiu=(kayit *)malloc(sizeof(kayit));
for(er=0;er<s;++er)
{
lseek(handle1,er*sizeof(kayit),0);
read(handle1,ikiu,sizeof(kayit));
 WINDOWA0();
gotoxy(1,lor+1);
printf("|%-3d|%-8s|%-17s|%-9s|%-12s|%-14s|%-8s|\n",er+1,ikiu->sicno,ikiu->isim,ikiu->telno,
kiu->cad,ikiu->dyer,ikiu->dtar);
lor++;
}
printf("|                                                                                |\n");
er=qyer(1,er);gotoxy(1,er);printf("==>");
delline();sleep(1);
strcpy(aw->sio,ikiu->sicno);
wj=has(aw->sio);
xs=wj;
hes(xs,y);
vop=ayir(rt,jit);
vop=mu(vop,bh);
x[vop]=head[vop];
head[vop]=del(head[vop],aw->sio);
if(ur==0){
menu();
}
sa=sa+1;
gi=(yii *)malloc(sizeof(yii *));
gi->sii=sa;
fseek(stream0,0L,0);
write(handle0,gi,sizeof(yii));
lseek(handle1,(er-1)*sizeof(kayit),0);
read(handle1,rii0,sizeof(kayit));
lseek(handle4,(sa-1)*sizeof(kayit),0);
write(handle4,rii0,sizeof(kayit));
for(ki=er+1;ki<=s;++ki)
{
lseek(handle1,(ki-1)*sizeof(kayit),0);
read(handle1,cikis,sizeof(kayit));
```

```
cikis->ksay=ki-1;
lseek(handle1,(ki-2)*sizeof(kayit),0);
write(handle1,cikis,sizeof(kayit));
}
s=s-1;
for(di=1;di<=s;++di)
{
lseek(handle1,(di-1)*sizeof(kayit),0);
read(handle1,cikis,sizeof(kayit));
siram[di-1]->in=di;
strcpy(siram[di-1]->isim,cikis->isim);
tf[di-1]->tn=di;
strcpy(tf[di-1]->telno,cikis->telno);
dyeri[di-1]->dy=di;
strcpy(dyeri[di-1]->dyer,cikis->dyer);
res[di-1]->dr=di;
strcpy(res[di-1]->cad,cikis->cad);
ahia[di-1]->tnt=di;
strcpy(ahia[di-1]->onlet,cikis->dtar);
het[di-1]->k2=di;
strcpy(het[di-1]->sicno2,cikis->sicno);
}
k=s-1;
 sort(siram,0,s-1);
sirt(tf,0,s-1);s4(ahia,0,s-1);dog(dyeri,0,s-1);nor(res,0,s-1);
snt(het,0,s-1);
g->r=s;
write(handle2,g,sizeof(yi));
indoku();
ilk(); }
 break;
 case 17:
{
ur=1;
ri9=(kayit *)malloc(sizeof(kayit));
so();
gotoxy(40,18);
input(aw->sio,8);
wj=has(aw->sio);
xs=wj;
hes(xs,y);
vop=ayir(rt,jit);
vop=mu(vop,bh);
```

```
x[vop]=head[vop];
head[vop]=del(head[vop],aw->sio);
if(ur==0){
menu();
}
lseek(handle1,(zet-1)*sizeof(kayit),0);
read(handle1,ri9,sizeof(kayit));
baslik();
fas(ri9);
sa=sa+1;
g1=(yi1 *)malloc(sizeof(yi1 *));
g1->si1=sa;
fseek(stream0,0L,0);
write(handle0,g1,sizeof(yi1));
lseek(handle4,(sa-1)*sizeof(kayit),0);
write(handle4,ri9,sizeof(kayit));
for(ki=zet+1;ki<=s;++ki)
{
lseek(handle1,(ki-1)*sizeof(kayit),0);
read(handle1,cikis,sizeof(kayit));
cikis->ksay=ki-1;
lseek(handle1,(ki-2)*sizeof(kayit),0);
write(handle1,cikis,sizeof(kayit));
}
s=s-1;
for(di=1;di<=s;++di)
{
lseek(handle1,(di-1)*sizeof(kayit),0);
read(handle1,cikis,sizeof(kayit));
siram[di-1]->in=di;
strcpy(siram[di-1]->isim,cikis->isim);
tf[di-1]->tn=di;
strcpy(tf[di-1]->telno,cikis->telno);
dyeri[di-1]->dy=di;
strcpy(dyeri[di-1]->dyer,cikis->dyer);
res[di-1]->dr=di;
strcpy(res[di-1]->cad,cikis->cad);
ahia[di-1]->tnt=di;
strcpy(ahia[di-1]->onlet,cikis->dtar);
het[di-1]->k2=di;
strcpy(het[di-1]->sicno2,cikis->sicno);
}
k=s-1;
```

```
 sort(siram,0,s-1); ant(het,0,s-1);
sirt(ttf,0,s-1);s4(ahia,0,s-1);dog(dyeri,0,s-1);nor(res,0,s-1);
g->r=s;
write(handle2,g,sizeof(yi));
indoku();
ilk();
}
break;
case 19:
{
menu();
}
break;
}
to=to+1;
if (to%2==0 )ilk();
fclose(stream1);
fclose(stream2);
fclose(stream4);
fclose(stream0);
menu();
}
struct bucket *del(sir,cop)
struct bucket *sir;
sal *cop;
{
int f,gh;
int handle1;
struct bucket *lk,*son;
lk=sir;
while(strcmp(sir->sicno1,cop->sio)!=0 && sir->link !=NULL)
{
son=sir;
sir=sir->link;
}
if(strcmp(sir->sicno1,cop->sio)==0)
{zet=sir->k1;
if(strcmp(lk->sicno1,cop->sio)==0)
{
lk=sir->link;
}
else {
son->link=sir->link;
free(sir);
}
```

```
}
else
{
ur=0;printf("ARADIGINIZ KAYIT YOK\n");
}
return(ik);
}
struct bucket *ara(sir,cop)
struct bucket *sir;
sal *cop;
{
int f,gh;
int handle1;
FILE *stream1;
struct bucket *ik,*son;
kayit *ri8;
ri8=(kayit *)malloc(sizeof(kayit *));
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
while(strcmp(sir->sicno1,cop->sio)!=0 && sir->link !=NULL)
{
sir=sir->link;
}
if(strcmp(sir->sicno1,cop->sio)==0)
{
baslik();
zet=sir->k1;
lseek(handle1,(zet-1)*sizeof(kayit),0);
read(handle1,ri8,sizeof(kayit));
fas(ri8);
getch();
}
else
{
printf("ARADIGINIZ KAYIT YOK\n");
ur=0;
}
}
mu(vop,bh)
{
if(vop(bh)
{
vop=ayir(rt,jit+1);
```

```
}
else{
 vop=ayir(rt,jit);
}
return(vop);
}


yazi()
{
int fi=0;
WINDOW002();clrscr();
ekryaz();
while(fi<ma)
{
xi[fi]=eadi[fi];
dii(xi[fi]);
fi++;getch();
}
}
dii(opi)
struct ket *opi;
{
int k3i;
while(opi!=NULL) {
opi=opi->inki;
}
WINDOW002();
}

ilk()
{
int q2,bn,c;
int handlei;
s=dosoku();
st=lf*bkfr;
ma=s/st;
for(q2=0;q2<ma;++q2)
{
x[q2]=NULL;
}
for(q2=0;q2<ma;++q2)
{
xi[q2]=NULL;
```

```
}
jit=hesap(ma);
bn=us(2,jit);
bh=ma-bn;
ndex();
nd1();
nd4();
nd5();nd6();nd7();
}

hes(aa,bb)
{
int mask=1,loc;
for(loc=0;mask !=0;++loc)
{
bb=bb-1;
if(aa&mask) rt[bb]='1';
else
rt[bb]='0';
mask=mask<<1;
}
}
hesap(ma)
{
int l=0,bv=0,ny;
while(bv<=ma){
l++;bv=us(2,l);
}
return(l-1);
}

ayir(rt,xy)
char rt[16];
int xy;
{
int vb,cv,top=0;
for(vb=0;vb<=xy-1;++vb)
{
if(rt[15-vb]=='1')
{
cv=us(2,vb);
top=top+cv;}
}
return(top);
```

```
}
has(snn)
char *snn;
{int ai=0;
int aa=0;
while(*(snn+ai) !='\0'){
aa+=a(snn+ai);
ai++;
}
return(aa);
}
us(jj,oo)
int jj;
int oo;
{
int hh=1,zz;
for(zz=1;zz<=oo;++zz)
{
hh=hh*jj;
}
return(hh);
}
ndi()
{
int j,xs,klo=0;
while(klo<s)
{
j=has(siram[klo]->isim);
xs=j;
hes(xs,y);
vop=ayir(rt,jit);
mul(vop,bh);
if(xi[vop]==NULL){
eadi[vop]=xi[vop]=(struct ket *)malloc(sizeof(struct ket));
strcpy(xi[vop]->is3,siram[klo]->isim);
xi[vop]->k3=siram[klo]->in;
xi[vop]->ink1=NULL;
}
else{
xi[vop]=ti(siram[klo],xi[vop]);
}
klo++;
}
}
```

```
struct ket *t1(xx1,ac1)
struct ket *ac1;
is *xx1;
{
if(ac1==NULL)
ac1=(struct ket *)malloc(sizeof(struct ket));
strcpy(ac1->is3,xx1->isim);
ac1->k3=xx1->in;
ac1->ink1=NULL;
}
else
ac1->ink1=t1(xx1,ac1->ink1);
return(ac1);
}
nd6()
{
int j,xs,klo=0;
while(klo<s)
{
j=has(dyeri[klo]->dyer);
xs=j;
hes(xs,y);
vop=a,ir(rt,jit);
mu(vop,bh);
if(x6[vop]==NULL){
ead6[vop]=x6[vop]=(struct ket6 *)malloc(sizeof(struct ket6));
strcpy(x6[vop]->dy6,dyeri[klo]->dyer);
x6[vop]->k6=dyeri[klo]->dy;
x6[vop]->ink6=NULL;
}
elsei
x6[vop]=t6(dyeri[klo],x6[vop]);
}
klo++;
}
}

struct ket6 *t6(xx6,ac6)
struct ket6 *ac6;
yer *xx6;
{
if(ac6==NULL)
ac6=(struct ket6 *)malloc(sizeof(struct ket6));
strcpy(ac6->dy6,xx6->dyer);
ac6->k6=xx6->dy;
```

```
ac6->ink6=NULL;
}
else
ac6->ink6=t6(xx6,ac6->ink6);
return(ac6);
}
nd7()
{
int j,xs,klo=0;
while(klo<s)
{
j=has(trh[klo]->dtar);
xs=j;
hes(xs,y);
vop=ayir(rt,jit);
mu(vop,bh);
if(x7[vop]==NULL){
ead7[vop]=x7[vop]=(struct ket7 *)malloc(sizeof(struct ket7));
strcpy(x7[vop]->dt7,ahia[klo]->onlet);
x7[vop]->k7=ahia[klo]->tnt;
x7[vop]->ink7=NULL;
}
else{
x7[vop]=t7(ahia[klo],x7[vop]);
}
klo++;
}
}

struct ket7 *t7(xx7,ac7)
struct ket7 *ac7;
not *xx7;
{
if(ac7==NULL)
{ac7=(struct ket7 *)malloc(sizeof(struct ket7));
strcpy(ac7->dt7,xx7->onlet);
ac7->k7=xx7->tnt;
ac7->ink7=NULL;
}
else
ac7->ink7=t7(xx7,ac7->ink7);
return(ac7);
}

nd4()
```

```
{
int j,xs,klo=0;
while(klo<s)
{
j=has(tf[klo]->telno);
xs=j;
hes(xs,y);
vop=ayi(rt,jit);
mu(vop,bh);
if(x4[vop]==NULL){
ead4[vop]=x4[vop]=(struct ket4 *)malloc(sizeof(struct ket4));
strcpy(x4[vop]->tel4,tf[klo]->telno);
x4[vop]->k4=tf[klo]->tn;
x4[vop]->ink4=NULL;
}
else{
x4[vop]=t4(tf[klo],x4[vop]);
}
klo++;
}
}

struct ket4 *t4(xx4,ac4)
struct ket4 *ac4;
no *xx4;
{
if(ac4==NULL)
{ac4=(struct ket4 *)malloc(sizeof(struct ket4));
strcpy(ac4->tel4,xx4->telno);
ac4->k4=xx4->tn;
ac4->ink4=NULL;
}
else
ac4->ink4=t4(xx4,ac4->ink4);
return(ac4);
}
nd5()
{
int j,xs,klo=0;
while(klo<s)
{
j=has(res[klo]->cad);
xs=j;
hes(xs,y);
```

```
vop=ayir(rt,jit);
mu(vop,bh);
if(x5[vop]==NULL){
ead5[vop]=x5[vop]=(struct ket5 *)malloc(sizeof(struct ket5));
strcpy(x5[vop]->ca5,res[klo]->cad);
x5[vop]->k5=tf[klo]->tn;
x5[vop]->ink5=NULL;
}
else{
x5[vop]=t5(res[klo],x5[vop]);
}
klo++;
}
}

struct ket5 *t5(xx5,ac5)
struct ket5 *ac5;
ad *xx5;
{
if(ac5==NULL)
{ac5=(struct ket5 *)malloc(sizeof(struct ket5));
strcpy(ac5->ca5,xx5->cad);
ac5->k5=xx5->dr;
ac5->ink5=NULL;
}
else
ac5->ink5=t5(xx5,ac5->ink5);
return(ac5);
}

ndex()
{
int j,xs,klo=0;
while(klo<s)
{
j=has(het[klo]->sicno2);
xs=j;
hes(xs,y);
vop=ayir(rt,jit);
mu(vop,bh);
if(x[vop]==NULL){
head[vop]=x[vop]=new(x[vop]);
strcpy(x[vop]->sicno1,het[klo]->sicno2);
x[vop]->k1=het[klo]->k2;
```

```
x[vop]->link=NULL;
}
else{
x[vop]=ekle(het[klo],x[vop]);
}
klo++;
}
}
struct bucket *ekle(xx,ac)
struct bucket *ac;
yit *xx;
{
if(ac==NULL)
{ac=new(ac);
strcpy(ac->sicno1,xx->sicno2);
ac->k1=xx->k2;
ac->link=NULL;
}
else
ac->link=ekle(xx,ac->link);
return(ac);
}

struct bucket *new(xo)
struct bucket *xo;
{
xo=AL1;
return(xo);
}
degistirme()
{
int handle1,ij,w,v,l,r,q,lei,ya,cc=9,t=15;
int dfi,e5,a9,aj,axs,p1=0,p2=0,p3=0,p4=0,p5=0,p6=0;
char o[1],o2[17];
char c,*x1=" ";
char aya[11];char o5[16],o6[9];
kayit *ris,*iis;
FILE *stream1;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
ris=(kayit *)malloc(sizeof(kayit));
WINDOW002();clrscr();
xz3();
for(e5=0;e5<s;++e5)
{
```

```
lseek(handle1,e5*sizeof(kayit),0);
read(handle1,ris,sizeof(kayit));
printf("|%-3d|%-8s|%-17s|%-9s|%-12s|%-14s|%-8s|\n",ris->ksay,ris->sicno,ris->isim,ris->telno
,ris->cad,ris->dyer,ris->dtar);
}
printf("|                                                                                |\n");
ya=qyer(1,4);gotoxy(1,ya);printf("==>");sleep(1);
lseek(handle1,(ya-1)*sizeof(kayit),0);
read(handle1,ris,sizeof(kayit));
iis=(kayit *)malloc(sizeof(kayit));
WINDOW002();
clrscr();
baslik();
eg();
fas(ris);
gotoxy(25,15);
do{
t=qyer(25,t);
switch(t)
{
case 15:
{
gotoxy(25,t);
for(ij=0;ij<8;++ij)
{
*(si+ij)=getche();*(si+ij+1)=0;
}
gotoxy(25,16);
strcpy(iis->sicno,si);
free(si);
p1=1;
}
break;
case 16:
{
gotoxy(25,t);
for(r=0;r<15;++r)
{
*(iasim+r)=getche();*(iasim+r+1)=0;
}
gotoxy(25,17);
strcpy(iis->isim,iasim);
free(iasim);
p2=1;
}
break;
```

```
case 17:
{
gotoxy(25,t);
for(w=0;w<9;++w)
{
*(tan+w)=getche();*(tan+w+1)=0;
}
gotoxy(25,18);
strcpy(iis->telno,tan);
free(tan);
p3=1;
}
break;
case 18:
{
gotoxy(25,t);
for(v=0;v<11;++v)
{
*(caad+v)=getche();*(caad+v+1)=0;
}
gotoxy(25,19);
strcpy(iis->cad,caad);
free(caad);
p4=1;
}
break;
case 19:
{
gotoxy(25,t);
for(l=0;l<13;++l)
{
*(dayer+l)=getche();*(dayer+l+1)=0;
}
gotoxy(25,20);
strcpy(iis->dyer,dayer);
free(dayer);
p5=1;
}
break;
case 20:
{
gotoxy(25,t);
for(q=0;q<8;++q)
{
```

```
(dat+q)=getche();*(dat+q+1)=0;
}
gotoxy(25,21);
strcpy(iis->dtar,dat);
free(dat);
p6=1;
}
break;
}
*x1=getch();
cc=*(x1);
}while(cc!=27);
if(p1==0)
{
strcpy(iis->sicno,ris->sicno);
iis->ksay=ya;
}
if(p2==0)
{
strcpy(iis->isim,ris->isim);
}
if(p3==0)
{
strcpy(iis->telno,ris->telno);
}
if(p4==0)
{
strcpy(iis->cad,ris->cad);
}
if(p5==0)
{
strcpy(iis->dyer,ris->dyer);
}
if(p6==0)
{
strcpy(iis->dtar,ris->dtar);
}
 iis->ksay=ya;
lseek(handle1,(ya-1)*sizeof(kayit),0);
write(handle1,iis,sizeof(kayit));
for(ya=1;ya<=s;++ya)
{
lseek(handle1,(ya-1)*sizeof(kayit),0);
read(handle1,iis,sizeof(kayit));
```

```
siram[ya-1]->in=ya;
strcpy(siram[ya-1]->isim,iis->isim);
tf[ya-1]->tn=ya;
strcpy(tf[ya-1]->telno,iis->telno);
dyeri[ya-1]->dy=ya;
strcpy(dyeri[ya-1]->dyer,iis->dyer);
res[ya-1]->dr=ya;
strcpy(res[ya-1]->cad,iis->cad);
ahia[ya-1]->tnt=ya;
strcpy(ahia[ya-1]->onlet,iis->dtar);
het[ya-1]->k2=ya;
strcpy(het[ya-1]->sicno2,iis->sicno);
}
 sort(siram,0,s-1); snt(het,0,s-1);
sirt(tf,0,s-1);s4(ahia,0,s-1);dog(dyeri,0,s-1);nor(res,0,s-1);
indoku();
ilk();
fclose(stream1);
}

struct ket *ra(iir,iop)
struct ket *iir;
char *iop;
{
int handle1;
FILE *stream1;
struct ket *jlk,*jon;
kayit *ri9;
ri9=(kayit *)malloc(sizeof(kayit *));
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
jon=iir->ink1;
while(strcmp(iir->is3,iop)!=0 && iir->ink1 !=NULL)
{
iir=jon;
jon=jon->ink1;
}
if(strcmp(iir->is3,iop)==0)
{
baslik();
iet=iir->k3;
lseek(handle1,(iet-1)*sizeof(kayit),0);
read(handle1,ri9,sizeof(kayit));
fas(ri9);
```

```
getch();
}
else
printf("ARADIGINIZ KAYIT YOK\n");
}

kayit_arama()
{
int handle1,handle2;
int xs,wij,sij,ii,mia,vcd,ij,rm,vf,q,b,ti,maxlen;
kayit *hf,*orf,*horf,*rf,*arf;
char *ade,dc,jh,si0[16];
FILE *stream1,*stream3;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);

do{
  WINDOW002();clrscr();
printf("                                        \n");
printf("            ARAMA MENUSU                \n");
printf("                                        \n");
printf("          <  >SICIL NOYA  GORE          \n");
printf("          <  >ISME GORE                 \n");
printf("          <  >HARFLERE GORE             \n");
printf("          <  >CIKIS                     \n");
printf("                                        \n");
printf("                                        \n");
dc=qyer(11,8);gotoxy(11,dc);
printf("=>");sleep(1);
if(dc==7) menu();
if(dc==4)
{
WINDOW002();clrscr();
so();gotoxy(40,18);
input(aw->sio,8);
wij=has(aw->sio);
xs=wij;
hes(xs,y);
vop=ayir(rt,jit);
vop=mu(vop,bh);
ara(head[vop],aw->sio);
}
if(dc==5)
{
```

```
if(strncmp(dyeri[aij]->dyer,kya,maxlen)==0)
{
rm=dyeri[aij]->dy;
lseek(handle1,(rm-1)*sizeof(kayit),0);read(handle1,orf,sizeof(kayit));
gotoxy(1,(aij+1));printf("%-3d|%-14s|%-8s|%-17s|%-9s|%-12s|%-8s|\n",aij+1,orf->dyer,
orf->sicno,orf->isim,orf->telno,orf->cad,orf->dtar);
}
}
}
if(vcd==8)
{ WINDOW002();clrscr();
horf=(kayit *)malloc(sizeof(kayit));
printf("bas harfleri yaziniz=");
gets(kya);
maxlen=strlen(kya);
 WINDOW002();clrscr();
xz2();
for(aij=0;aij<s;++aij)
{
if(strncmp(ahia[aij]->onlet,kya,maxlen)==0)
{
rm=ahia[aij]->tnt;
lseek(handle1,(rm-1)*sizeof(kayit),0);read(handle1,horf,sizeof(kayit));
printf("%-3d|%-8s|%-8s|%-17s|%-9s|%-12s|%-14s|\n",aij+1,horf->dtar,horf->sicno,
horf->isim,horf->telno,horf->cad,horf->dyer);
}
}
}
}
printf("\n\n\n\n\tDEVAM<D>----MENU<M>");jh=getch();
}while(jh=='d' || jh=='D');
fclose(stream1);
}
```

```
WINDOW002();clrscr();
gotoxy(1,2);printf("\n\tismi giriniz");gets(si0);
ism=(kayit *)malloc(sizeof(kayit));
wij=has(si0);
xs=wij;
hes(xs,y);
vop=ayir(rt,jit);
vop=mu(vop,bh);
ra(eadl[vop],si0);
}
if(dc==6)
{   indoku();
WINDOW002();clrscr();
printf("                                        \n");
printf("       HARFLERLE ARAMA MENUSU           \n");
printf("                                        \n");
printf("       <   >isme gore                   \n");
printf("       <   >tel noya gore               \n");
printf("       <   >adrese gore                 \n");
printf("       <   >dogum yerine gore           \n");
printf("       <   >dogum tarihine gore         \n");
printf("       <   >                            \n");
printf("                                        \n");
vcd=qyer(10,10);gotoxy(10,vcd);printf("==>");
sleep(1);
if(vcd==4)
{WINDOW002();clrscr();
printf("bas harfleri yaziniz=");
gets(kya);
maxlen=strlen(kya);
 WINDOW002();clrscr();
ekryaz();
hf=(kayit *)malloc(sizeof(kayit));
for(aij=0;aij<s;++aij)
{
if(strncmp(siram[aij]->isim,kya,maxlen)==0)
{
rm=siram[aij]->in;
lseek(handle1,(rm-1)*sizeof(kayit),0);read(handle1,hf,sizeof(kayit));
printf("|%-3d|%-16s|%-8s|%-9s|%-12s|%-14s|%-9s|\n",(aij+1),hf->isim,hf->sicno,hf->telno,
hf->cad,hf->dyer,hf->dtar);
}
}
}
if(vcd==5)
{WINDOW002();clrscr();
printf("bas harfleri yaziniz=");
```

```
gets(kya);
maxlen=strlen(kya);
 WINDOW002();clrscr();
ez();
arf=(kayit *)malloc(sizeof(kayit));
for(aij=0;aij<s;++aij)
{
if(strncmp(tf[aij]->telno,kya,maxlen)==0)
{
rm=tf[aij]->tn;
lseek(handle1,(rm-1)*sizeof(kayit),0);read(handle1,arf,sizeof(kayit));
printf("|%-3d|%-9s|%-8s|%-16s|%-12s|%-14s|%-9s|\n",aij+1,arf->telno,arf->sicno,arf->isim,
arf->cad,arf->dyer,arf->dtar);
}
}
}
if(vcd==6)
{ WINDOW002();clrscr();
printf("bas harfleri yaziniz=");
gets(kya);
maxlen=strlen(kya);
 WINDOW002();clrscr();
xz();
rf=(kayit *)malloc(sizeof(kayit));
for(aij=0;aij<s;++aij)
{
if(strncmp(res[aij]->cad,kya,maxlen)==0)
{
rm=res[aij]->dr;
lseek(handle1,(rm-1)*sizeof(kayit),0);read(handle1,rf,sizeof(kayit));
printf("|%-3d|%-12s|%-8s|%-17s|%-9s|%-14s|%-8s|\n",aij+1,rf->cad,rf->sicno,rf->isim,
rf->telno,rf->dyer,rf->dtar);
}
}
}
if(vcd==7)
{ WINDOW002();clrscr();
printf("bas harfleri yaziniz=");
gets(kya);
maxlen=strlen(kya);
 WINDOW002();clrscr();
xz1();
orf=(kayit *)malloc(sizeof(kayit));
for(aij=0;aij<s;++aij)
{
```

```
dossil()
{
char cgt;

WINDOW002();
clrscr();
printf("\tEMINMISINIZ ..\n");
printf("\t(evet icin  (e)   tusuna basiniz)=\n");
cgt=getche();
if (cgt!='e') return;
printf("\n\t\tDOSYALAR SILINIYOR....\n");
/*unlink(dya);*/
remove("sayi");
remove("si");
remove("bal");
remove("index");
remove("adres");
remove("telno");
remove("dogtar");
remove("dogu");
remove("silen");
remove("sno");
}

indoku()
{
int handle3,handle5,handle6,handle7;
int d7,handle8,handle9;
FILE *stream3,*stream5,*stream6,*stream7;
FILE *stream8,*stream9;
int aid1,ie,igge,idge,id1,didge,did1;
stream3=fopen("index","r+b");
handle3=fileno(stream3);
stream5=fopen("telno","r+b");
handle5=fileno(stream5);
stream6=fopen("dogu","r+b");
handle6=fileno(stream6);
stream7=fopen("adres","r+b");
handle7=fileno(stream7);
stream8=fopen("dogtar","r+b");
handle8=fileno(stream8);
stream9=fopen("sno","r+b");
handle9=fileno(stream9);
for(ie=0;ie<s;++ie)
{siram[ie]=(is *)malloc(sizeof(is));
lseek(handle3,ie*sizeof(is),0);
read(handle3,siram[ie],sizeof(is));
 inim[ie]=siram[ie];
 }
```

```
for(igge=0;igge<s;++igge)
{     tf[igge]=(no *)malloc(sizeof(no));
lseek(handle5,igge*sizeof(no),0);
read(handle5,tf[igge],sizeof(no));
}
getch();
for(idge=0;idge<s;++idge)
{dyeri[idge]=(yer *)malloc(sizeof(yer));
lseek(handle6,idge*sizeof(yer),0);
read(handle6,dyeri[idge],sizeof(yer));
}
for(id1=0;id1<s;++id1)
{res[id1]=(ad *)malloc(sizeof(ad));
lseek(handle7,id1*sizeof(ad),0);
read(handle7,res[id1],sizeof(ad));
}
for(didge=0;didge<s;++didge)
{ahia[didge]=(not *)malloc(sizeof(not));
lseek(handle8,didge*sizeof(not),0);
read(handle8,ahia[didge],sizeof(not));
}
for(d7=0;d7<s;++d7)
{het[d7]=(yit *)malloc(sizeof(yit));
lseek(handle9,d7*sizeof(yit),0);
read(handle9,het[d7],sizeof(yit));
}
fclose(stream3);
fclose(stream5);
fclose(stream6);
fclose(stream7);
fclose(stream8);
fclose(stream9);
}
```

```
sort(siram,lt,rh)
is *siram[ks];
int lt,rh;
{
int handle3;
FILE *stream3;is *buf;
int f=0,l=0,fi;
char *x;
stream3=fopen("index","r+b");
handle3=fileno(stream3);
buf=(is *)malloc(sizeof(is));
f=lt;
l=rh;
x=siram[(lt+rh)/2]->isim;
do{
while(strcmp(siram[f]->isim,x)<0 && f<rh) f++;
while(strcmp(siram[l]->isim,x)>0 && l>lt) l--;
if(f<=l)
{
buf=siram[f];
siram[f]=siram[l];
siram[l]=buf;
f++;
l--;
}
}while(f<=l);
if(lt<l)sort(siram,lt,l);
if(f<rh)sort(siram,f,rh);
for(fi=0;fi<s;++fi){
lseek(handle3,fi*sizeof(is),0);
write(handle3,siram[fi],sizeof(is));
}
fclose(stream3);
}


snt(het,lnt,rnh)
yit *het[ks];
int lnt,rnh;
{
int handle9;
FILE *stream9;yit *bn8;
int fn=0,ln=0,fni;
```

```
char *xn;
stream9=fopen("sno","r+b");
handle9=fileno(stream9);
bn8=(yit *)malloc(sizeof(yit));
fn=lnt;
ln=rnh;
xn=het[(lnt+rnh)/2]->sicno2;
do{
while(strcmp(het[fn]->sicno2,xn)<0 && fn<rnh) fn++;
while(strcmp(het[ln]->sicno2,xn)>0 && ln>lnt) ln--;
if(fn<=ln)
{
bn8=het[fn];
het[fn]=het[ln];
het[ln]=bn8;
fn++;
ln--;
}
}while(fn<=ln);
if(lnt<ln)snt(het,lnt,ln);
if(fn<rnh)snt(het,fn,rnh);
for(fni=0;fni<s;++fni){
lseek(handle9,fni*sizeof(yit),0);
write(handle9,het[fni],sizeof(yit));
}
fclose(stream9);
}
dog(dyeri,lef,rig)
yer *dyeri[ks];
int lef,rig;
{
int handle6;
FILE *stream6; yer *dbuf;
int ddf=0,ddl=0,dfi;
char *ddx;
stream6=fopen("dogu","r+b");
handle6=fileno(stream6);
dbuf=(yer *)malloc(sizeof(yer));
ddf=lef;
ddl=rig;
ddx=dyeri[(lef+rig)/2]->dyer;
do{
while(strcmp(dyeri[ddf]->dyer,ddx)<0 && ddf<rig) ddf++;
while(strcmp(dyeri[ddl]->dyer,ddx)>0 && ddl>lef) ddl--;
```

```
if(ddf<=ddl)
{
dbuf=dyeri[ddf];
dyeri[ddf]=dyeri[ddl];
dyeri[ddl]=dbuf;
ddf++;
ddl--;
}
}while(ddf<=ddl);
if(lef<ddl)dog(dyeri,lef,ddl);
if(ddf<rig)dog(dyeri,ddf,rig) ;
for(dfi=0;dfi<s;++dfi){
lseek(handle6,dfi*sizeof(yer),0);
write(handle6,dyeri[dfi],sizeof(yer));
}
fclose(stream6);
}

sirt(tf,sol,sag)
no *tf[ks];
int sol,sag;
{
int handle5;
FILE *stream5;
no *ufo;
char *ta5;
int af=0,lz=0,fai;
ufo=(no *)malloc(sizeof(no));
stream5=fopen("telno","r+b");
handle5=fileno(stream5);
af=sol;
lz=sag;
ta5=tf[(sol+sag)/2]->telno;
do{
while(strcmp(tf[af]->telno,ta5)<0 && af<sag) af++;
while(strcmp(tf[lz]->telno,ta5)>0 && lz>sol) lz--;
if(af<=lz)
{
ufo=tf[af];
tf[af]=tf[lz];
tf[lz]=ufo;
af++;
lz--;
}
}while(af<=lz);
if(sol<lz)sirt(tf,sol,lz);
if(af<sag)sirt(tf,af,sag);
for(fai=0;fai<s;++fai){
lseek(handle5,fai*sizeof(no),0);
write(handle5,tf[fai],sizeof(no));
}
fclose(stream5);
}
```

```
s4(ahia,ol,ag)
not *ahia[ks];
int ol,ag;
{
int handle8;
char *ugno;
FILE *stream8;
not *nufo;
int naf=0,nlz=0,nfai;
nufo=(not *)malloc(sizeof(not));
stream8=fopen("dogtar","r+b");
handle8=fileno(stream8);
naf=ol;
nlz=ag;
ugno=ahia[(ol+ag)/2]->onlet;
do{
while(strcmp(ahia[naf]->onlet,ugno)>0 && naf<ag) naf++;
while(strcmp(ahia[nlz]->onlet,ugno)<0 && nlz>ol) nlz--;
if(naf<=nlz)
{
nufo=ahia[naf];
ahia[naf]=ahia[nlz];
ahia[nlz]=nufo;
naf++;
nlz--;
}
}while(naf<=nlz);
if(ol<nlz)s4(ahia,ol,nlz);
if(naf<ag)s4(ahia,naf,ag);
for(nfai=0;nfai<s;++nfai){
lseek(handle8,nfai*sizeof(not),0);
write(handle8,ahia[nfai],sizeof(not));
}
fclose(stream8);
}

nor(res,rol,rag)
ad *res[ks];
int rol,rag;
{
int handle7;
char *fo;
FILE *stream7;
ad *rufo;
int raf=0,rlz=0,rfai;
rufo=(ad *)malloc(sizeof(ad));
stream7=fopen("adres","r+b");
handle7=fileno(stream7);
```

```
raf=rol;
rlz=rag;
fo=res[(rol+rag)/2]->cad;
do{
while(strcmp(res[raf]->cad,fo)<0 && raf<rag) raf++;
while(strcmp(res[rlz]->cad,fo)>0 && rlz>rol) rlz--;
if(raf<=rlz)
{
rufo=res[raf];
res[raf]=res[rlz];
res[rlz]=rufo;
raf++;
rlz--;
}
}while(raf<=rlz);
if(rol<rlz)nor(res,rol,rlz);
if(raf<rag)nor(res,raf,rag);;
for(rfai=0;rfai<s;++rfai){
lseek(handle7,rfai*sizeof(ad),0);
write(handle7,res[rfai],sizeof(ad));
}
fclose(stream7);
}

cik()
{clrscr();exit(1);
clrscr();
}

dosya_ac()
{
int handle2,handle0,s=0,sa=0;
char x;
FILE *stream0,*stream2,*stream1,*stream3,*stream4,*stream5;
FILE *stream6,*stream7;
FILE *stream8,*stream9;
WINDOW1();clrscr();WINDOW2();clrscr();
printf("\n\n\n\ilk defa mi giriyorsunuz");x=getche();
if (x=='h' || x=='H'){WINDOW2(); return;}
if (x!='e') {WINDOW2();  return;}
stream2=fopen("say1","wb");
handle2=fileno(stream2);
stream0=fopen("si","wb");
handle0=fileno(stream0);
write(handle0,&sa,2);
fclose(stream0);
write(handle2,&s,2);
fclose(stream2);
stream1=fopen("bal","wb");
stream3=fopen("index","wb");
stream4=fopen("silen","wb");
stream5=fopen("telno","wb");
stream6=fopen("dogu","wb");
stream7=fopen("adres","wb");
stream8=fopen("dogtar","wb");
```

```
stream9=fopen("sno","wb");
fclose(stream1);
fclose(stream3);
fclose(stream4);
fclose(stream5);
fclose(stream6);
fclose(stream7);
fclose(stream8);
fclose(stream9);
}


dosoku()
{
int handle2,y;
FILE *stream2;
stream2=fopen("sayi","r+b");
handle2=fileno(stream2);
g=(yi *)malloc(sizeof(yi));
read(handle2,g,sizeof(yi));
y=g->r;
return(y);
}
sioku()
{
int handle0,y1;
FILE *stream0;
yi1 *g1;
stream0=fopen("si","r+b");
handle0=fileno(stream0);
g1=(yi1 *)malloc(sizeof(yi1));
read(handle0,g1,sizeof(yi1));
y1=g1->si1;
return(y1);
}

cizgi()
{
int jk;
for(jk=1;jk<=70;jk++) printf("=");
}
baslik()
{
int tsi;
 WINDOW002();clrscr();
for(tsi=0;tsi<13;++tsi)printf("\n");
printf("                                                                \n");
printf("        SICIL  NOSU =[       ]                                  \n");
printf("        ADI SOYADI  =[               ]                          \n");
printf("        TELEFON NO  =[         ]                                \n");
printf("        ADRESI      =[           ]                              \n");
printf("        DOGUM YERI  =[             ]                            \n");
printf("        DOGUM TARIHI=[        ]                                 \n");
printf("                                                                \n");
printf("                                                                \n");
}
```

```
eg()
{
int tsi,ce=24,je=25;
WINDOW002();
for(tsi=0;tsi<21;++tsi)printf("\n");
printf("                                                    \n");
printf(" YUKARI:          ESC:CIKIS          ASAGI:         \n");
printf("                                                    \n");
gotoxy(9,23);printf("%c",ce);gotoxy(50,23);printf("%c",je);
gotoxy(25,15);
}
fas(az)
kayit *az;
{WINDOW002();
gotoxy(25,15);printf("%s",az->sicno);
gotoxy(25,16);printf("%s",az->isim);
gotoxy(25,17);printf("%s",az->telno);
gotoxy(25,18);printf("%s",az->cad);
gotoxy(25,19);printf("%s",az->dyer);
gotoxy(25,20);printf("%s",az->dtar);}


menu()
{
char ch;
int h;
clrscr();WINDOW002();clrscr();
WINDOW127();clrscr();
do{WINDOW002();
WINDOW3();cprintf("D");WINDOW126();clrscr();
cprintf("osya islemleri\n");
WINDOW4();cprintf("Inde");WINDOW04();clrscr();cprintf("X\n");
WINDOW5();cprintf("K");WINDOW13();clrscr();cprintf("ayit\n");
WINDOW6();cprintf("G");WINDOW14();clrscr();cprintf("ozlem\n");
WINDOW7();cprintf("A");WINDOW8();clrscr();cprintf("rama");
WINDOW9();cprintf("S");WINDOW10();clrscr();cprintf("ilme");
WINDOW11();cprintf("s");WINDOW15();clrscr();cprintf("l");
WINDOW16();cprintf("ralama");
WINDOW17();cprintf("d");WINDOW18();cprintf("E");WINDOW19();cprintf("gistirme");
WINDOW22();cprintf("U");WINDOW23();clrscr();cprintf("ndelete");
WINDOW224();clrscr();
WINDOW24();clrscr();cprintf("F1:CIKIS");
ch=getch();
```

```
cizi()
{int i;
for(i=1;i<=76;++i)
printf("_");
}
silme()
{
struct ffblk ffblk;
int yap,hu=1,khu=1;
/*char dosism;*/
WINDOW002();
clrscr();
/*scanf("%8s",&dosism);*/
printf("\t\t\t\tDOSYALAR OKUNUYOR.....\n");
yap=findfirst("*.*",&ffblk,0);
while(!yap)
{
printf("%s\n",ffblk.ff_name);hu++;
yap=findnext(&ffblk);
if((hu)==23*khu){printf("bir tusa basiniz....\n");getch();khu++;}
}
/*while(dosism!=ffblk.ff_name)
if(strcmp(dosism,ffblk.ff_name)=0)  */
printf("BIR TUSA BASINIZ");getch();
}

dosyaislemleri()
{
 char dosy;
 WINDOW002();clrscr();
printf("           DOSYALAMA MENUSU\n");
printf("          -----------------\n");
WINDOW3();cprintf("D");WINDOW12();clrscr();
cprintf("osya yaratma\n");
WINDOW5();cprintf("L");WINDOW13();clrscr();printf("iste\n");
WINDOW6();cprintf("S");WINDOW14();clrscr();printf("ilme \n");
dosy=getche();
switch(dosy)
{
case 'd':
case 'D':
{
dosya_ac();
menu();
}
```

```
break;
case 'l':
case 'L':
{
silme();
menu();
       }
break;
case 's':
case 'S':
{
dossil();
menu();
}
break;
}
}
index()
{
int handle1;
int kad,pn1=0,g3,pn3,adyq,ked,kq,o3,tkq,tked;
int sec,sao,kebe;
char fff,dd,jkec,rec,jh,gec;
FILE *stream1;
kayit *dx;
stream1=fopen("bal","r+b");
handle1=fileno(stream1);
do
{
WINDOW002(); clrscr();
sma();
ama();
sec=qyer(27,22);gotoxy(27,sec);printf("->");sleep(1);
switch(sec)
{
case 11:
{
WINDOW002();clrscr();
printf("                SICIL NOYA GORE INDEX                \n");
xz3();
dx=(kayit *)malloc(sizeof(kayit));
for(g3=0;g3<s;++g3)
{
```

```
xz1();
yerdi=(kayit *)malloc(sizeof(kayit));
for(adyq=0;adyq<s;++adyq)
{
kad=dyeri[adyq]->dy;
lseek(handle1,(kad-1)*sizeof(kayit),0);read(handle1,yerdi,sizeof(kayit));
printf("\n|%-3d|%-14s|%-8s|%-17s|%-9s|%-12s|%-8s|",kad,yerdi->dyer,yerdi->sicno,yerdi->isim,
yerdi->telno,yerdi->cad,yerdi->dtar);
}
printf("\n|\n");
WINDOW002();
}
break;
case 14:
{WINDOW002();   clrscr();
printf("\a|          ADRESE    GORE    INDEX          |\n");
xz();
cade=(kayit *)malloc(sizeof(kayit));
for(kq=0;kq<s;++kq)
{
ked=res[kq]->dr;
lseek(handle1,(ked-1)*sizeof(kayit),0);read(handle1,cade,sizeof(kayit));
printf("\n|%-3d|%-12s|%-8s|%-17s|%-9s|%-14s|%-8s|",ked,cade->cad,cade->sicno,cade->isim,
cade->telno,cade->dyer,cade->dtar);
}
printf("\n|\n");
WINDOW002();
}
break;
case 16:
{WINDOW002();   clrscr();
printf("\a|          DOGUM TARIHINE GORE INDEX          |\n");
/*tade=(kayit *)malloc(sizeof(kayit));*/
xz2();
for(tkq=0;tkq<s;++tkq)
{
tked=ahia[tkq]->tnt;
lseek(handle1,(tked-1)*sizeof(kayit),0);read(handle1,tade,sizeof(kayit));
printf("\n|%-3d|%-8s|%-8s|%-17s|%-9s|%-12s|%-14s|",tked,tade->dtar,tade->sicno,tade->isim,
tade->telno,tade->cad,tade->dyer);
}
printf("\n|\n");
WINDOW002();
}
```

```
pn3=het[g3]->k2;
lseek(handle1,(pn3-1)*sizeof(kayit),0);read(handle1,dx,sizeof(kayit));
printf("\n %-3d %-8s %-16s %-9s %-12s %-14s %-9s ",pn3,dx->sicno,dx->isim,dx->telno,dx->cad,
dx->dyer,dx->dtar);
}
printf("\n ═══════════════════════════════════════════════════════════════ \n");
WINDOW002();
}
break;
case 12:
{ WINDOW002();clrscr();
printf(" ═══════════ ISME   GORE INDEX ═══════════ \n");
ekryaz();
 reno=(kayit *)malloc(sizeof(kayit));
for(o3=0;o3<s;++o3)
{
pn1=siram[o3]->in;
lseek(handle1,(pn1-1)*sizeof(kayit),0);read(handle1,dex,sizeof(kayit));
printf("\n %-3d %-16s %-8s %-9s %-12s %-14s %-9s ",pn1,dex->isim,dex->sicno,dex->telno,
dex->cad,dex->dyer,dex->dtar);
}
printf("\n ═══════════════════════════════════════════════════════════ \n");
WINDOW002();
}
break;
case 13:
{
WINDOW002();clrscr();
printf(" ═══════════ TELEFON NOYA GORE INDEX ═══════════ \n");
ez();
reno=(kayit *)malloc(sizeof(kayit));
for(sao=0;sao<s;++sao)
{
kebe=tf[sao]->tn;lseek(handle1,(kebe-1)*sizeof(kayit),0);read(handle1,reno,sizeof(kayit));
printf("\n %-3d %-9s %-8s %-16s %-12s %-14s %-9s ",kebe,reno->telno,reno->sicno,reno->isim,
reno->cad,reno->dyer,reno->dtar);
}
printf("\n ═══════════════════════════════════════════════════════════ \n");
WINDOW002();
}
break;
case 15:
{         WINDOW002();clrscr();
printf("\a ═══════════ DOGUM YERINE GORE INDEX ═══════════ \n");
```

```
break;
}
gotoxy(1,24);printf("\t\t\tDEVAM<D>----MENU<M>");jh=getch();
}while(jh=='d' || jh=='D');
}
so()
{
clrscr();
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("                    ┌──────────────────────────────────┐\n");
printf("                    │                                  │\n");
printf("                    │        <->>SICIL NO=             │\n");
printf("                    │                                  │\n");
printf("                    │                                  │\n");
printf("                    └──────────────────────────────────┘\n");
}
ana()
{
int ce=24,je=25;
printf("\t\t┌──────────────────────────────────┐\n");
printf("\t\t│                                  │\n");
printf("\t\t│        <  >SICIL NO              │\n");
printf("\t\t│        <  >ADI SOYADI            │\n");
printf("\t\t│        <  >TELEFON NO            │\n");
printf("\t\t│        <  >ADRES                 │\n");
printf("\t\t│        <  >DOGUM YERI            │\n");
printf("\t\t│        <  >DOGUM TARIHI          │\n");
printf("\t\t│        <  >ANA MENUYE DON        │\n");
printf("\t\t│                                  │\n");
printf("\t\t│                                  │\n");
printf("\t\t│                                  │\n");
printf("\t\t├──────────────────────────────────┤\n");
printf("\t\t│                                  │\n");
printf("\t\t│YUKARI:                 ASAGI:    │\n");
printf("\t\t└──────────────────────────────────┘\n");
gotoxy(25,22);printf("%c",ce);gotoxy(56,22);printf("%c",je);
gotoxy(26,22);
}
sma()
{
clrscr();
printf("\n\n\n\n\n\n\n");
printf("\t\t┌──────────────────────────────────┐\n");
printf("\t\t│          INDEX    MENUSU         │\n");
}
```

Ek - B

Richard-P. Brent Metodunun Programlanması

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
char *s0;
int mode,key,keytab[5];
main()
{
int i;
char bn;
for(i=0;i<=5;i++)
 {
keytab[i]=0;
}
while(mode!=20)
{
has();
}
}
has()
{
int i;
int keyq[5],keyir[5],kvtab[5];
int ia,is,ix,jr,ic=-1,ir,iq;
int  len=5, len2= 3;
int del,x,y,jq,bu,kt,ka;
char a;
menu();

printf("%d",mode);
if(mode==19){mode=1;}
if(mode==18){mode=3;}
if(mode==17){mode=2;}
if(mode==2)
{
do{
key=random(14);
clrscr();
printf("KEY=%d",key);
getch();
}while(key==0);
}
if(mode==1)
{
```

```
clrscr();
printf("\t\nTABLODAKI KEYLER ");
printf("\t\n╔═══════════════╗");
for(i=0;i<5;i++)
 {
printf("\t\n║ %d          %-2d  ║",i,keytab[i]);
}
printf("\t\n╚═══════════════╝\n");
printf("\nARANILACAK KEY=");scanf("%d",&key);
}
if(mode==3)
{
clrscr();
printf("\t\nTABLODAKI KEYLER ");
printf("\t\n╔═══════════════╗");
for(i=0;i<5;i++)
 {
printf("\t\n║ %d          %-2d  ║",i,keytab[i]);
}
printf("\t\n╚═══════════════╝\n");
printf("\nSILINECEK KEY=");scanf("%d",&key);
}

if(mode==20)
{
exit(1);
}
x=key%len2;
iq=abs(x);
y=key%len;
ir=abs(y)+1;
ka=ir;
printf("ADRESI=%d",ka);
getch();
g20: kt=keytab[ka];
if (kt==0) goto g30;
if(kt==-1) goto g40;
if(kt==key) goto g60;
ic=ic+1;
ka=ka+iq;
if(ka>len)  ka=ka-len;
if(ka!=ir) goto g20;
g30:printf("\t\nKEY TABLODA YOK\n");
if(mode==2 && ic<=len2 && key !=0 && key != -1) {
 goto g70;
}
```

```
else
ka=0;
return;
g40: ia=ka;
g50: ia=ia+iq;
if(ia>len){ia=ia-len;}
is=keytab[ia];
if(is==0 || ia==ir ) goto g30;
if((is !=key ) || (is ==-1))  goto g50;
kvtab[ka]=is;
keytab[ia]=-1;
g60:
printf("\t\nKEY TABLODAN BULUNDU, TABLODAKI ADRESI= %d\n ",kt);
sleep(1);
if(mode==3)
 {
keytab[ka]=-1;
printf("TABLODAN SILINDI");
menu();
}
return;
 g70:if(ic<=0) goto g120;
 del=kt;
 ia=ka;
 is=0;
 g80: ix=ic-is;
 bu=keytab[ir];
 jq=bu%len2+1;
 jr=ir;
 g90: jr=jr+jq;
 if(jr>len){
 jr=jr-len ;
 }
 kt=keytab[jr];
 if((kt==0) || (kt==-1)) goto g100;
ix=ix-1;
if(ix>0) goto g90;
if((ix<0)||(ix==0)) goto g110;
g100: if(del!=0 && (kt==0)) goto g110;
if(kt !=0){printf("silinebilir");}
ia=jr;
ka=ir;
ic=ic-ix;
g110:is=is+1;
```

```
ir=ir+iq;
if(ir>len){ ir=ir-len;}
if(ic>is) goto g80;
if(ia==ka)goto g120;
kvtab[ia]=kvtab[ka];
keytab[ia]=keytab[ka];
g120: keytab[ka]=key;
printf("\nTABLOYA KAYDEDILDI");
getch();
return;
}
qyer(q2,q1)
int q1,q2;
{
int c=0;
char *s0=" ";
gotoxy(q2,q1);
while(c!=13)
{
*s0=getch();
c=*s0;
if(c==72){
gotoxy(q2,q1-1);q1=q1-1;
}
if(c==80){
gotoxy(q2,q1+1);q1=q1+1;
}
}
return(q1);
}
menu()
{
clrscr();
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("                                                    \n");
printf("                    ANA MENU                        \n");
printf("                                                    \n");
printf("                                                    \n");
printf("                  <  >KAYIT                         \n");
printf("                  <  >SILME                         \n");
printf("                  <  >ARAMA                         \n");
printf("                  <  >CIKIS                         \n");
printf("                                                    \n");
printf("                                                    \n");
mode=qyer(28,21);gotoxy(28,mode);printf("->");sleep(1);
}
```

ÖZGEÇMİŞ

1960 yılında Urfa'da doğdu. 1976-77 öğretim yılında İstanbul Şehremini Lisesini bitirdi 1979-83 öğretim yılları arasında İ.T.Ü. Matematik Mühendisliği bölümünde öğrenimini tamamladı. Bir süre programcı olarak Özel Sektörde çalıştıktan sonra M.S.Ü. Fen-Edebiyat Fakültesine Arş.Gör. olarak girdi. Halen görevine devam etmektedir.