T.C.
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

TEMEL ALGORİTMALAR

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

İÇİNDEKİLER i
AÇIKLAMALAR ii
GİRİŞ 1
ÖĞRENME FAALİYETİ-1 3
1. SIRALAMA 3
1.1. Ekleme Sıralaması 4
1.2. Balon Sıralaması 7
1.3. Kabuk Sıralaması 8
1.4. Hızlı Sıralama 10
1.5. Sıralama Algoritmaları 12
UYGULAMA FAALİYETİ 14
ÖLÇME VE DEĞERLENDİRME 15
ÖĞRENME FAALİYETİ-2 16
2. ARAMA 16
2.1. Sıralı (Ardışık) Arama 16
2.2. İkilik Arama 18
2.3. Kıyma (Hashing) Yöntemi 19
2.3.1. Kıyma Fonksiyonu 19
2.3.2. Kıyma Yönteminde Çakışmalar 20
2.3.3. Kıyma Yönteminde Arama 20
UYGULAMA FAALİYETİ 22
ÖLÇME VE DEĞERLENDİRME 23
ÖĞRENME FAALİYETİ-3 24
3. KOD İYİLEŞTİRME 24
3.1. Doğru Veri Yapısını Seçmek 24
3.2. Doğru Algoritmayı Seçmek 25
3.3. Kaynak Kodu İyileştirmek 25
UYGULAMA FAALİYETİ 29
ÖLÇME VE DEĞERLENDİRME 30
MODÜL DEĞERLENDİRME 31
CEVAP ANAHTARLARI 32
SÖZLÜK 33
KOD ÖRNEKLERİ 34
ÖNERİLEN KAYNAKLAR 42
KAYNAKÇA 43

AÇIKLAMALAR

KOD	**481BB0028**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Temel Algoritmalar**
MODÜLÜN TANIMI	Sıralama ve arama ile ilgili program yazımı, kodu iyileştirme ile ilgili öğrenme materyalidir.
SÜRE	40/24
ÖN KOŞUL	Veri Yapıları modülünü almış olmak.
YETERLİK	Temel algoritmalar yapmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, sıralama ve arama ile ilgili algoritmalar yazabilecek, programlamanın son aşaması olarak kodu iyileştirebileceksiniz. **Amaçlar** 1. Sıralama yapabileceksiniz. 2. Arama yapabileceksiniz. 3. Kodu iyileştirebileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan; bilgisayar, yazıcı, bilgisayar masaları, kâğıt, kalem, lisanslı işletim sistemi programı ve akış diyagramı sembolleri ile ilgili panolar.
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz. Sırasıyla araştırma yaparak, grup çalışmalarına katılarak ve en son aşamada alan öğretmenlerine danışarak ölçme ve değerlendirme uygulamalarını gerçekleştireceksiniz.

GİRİŞ

Sevgili Öğrenci,

Her şeyden önce herkes bir programlama dilini öğrenebilir. Bilgisayar programlama yüksek bir zekâ ve matematik bilgisi gerektirmez. Sadece asla vazgeçmeme sabrı ve öğrenme isteği yeterlidir.

Programlama bir hünerdir. Bazı insanlar doğal olarak diğerlerinden daha iyidir, ama herkes pratik yaparak iyi olabilir. Başaramamaktan korkmak yerine, kendinizi bu maharete vererek, öğrenmek için uğraşın. Programlama eğlencelidir, fakat yanlış çalışma yöntemleriyle sinir bozucu olabilir ve zamanınızın boşa geçmesine neden olabilir. Bu sebeple bu modülleri takip ederek, en az sıkıntı ve en yüksek memnuniyet ile programlamayı öğreneceksiniz.

Bu modül ile kazanacağınız bilgiler sıralama ve arama ile ilgili program yazmak ve kodu iyileştirme yöntemlerini kullanmaktır. Modülü bitirdiğinizde anlamadığınız yerleri tekrar okuyup, uygulayınız.

Bu modül programlama temelleri modüllerinin ilk dört modülünde gösterilen konuların bir araya gelmiş hâlidir. Takıldığınız yerlerde eski modüllerdeki bilgilere (diziler, veri yapıları, döngüler…) geri dönerek uygun konulara göz atınız.

ÖĞRENME FAALİYETİ–1

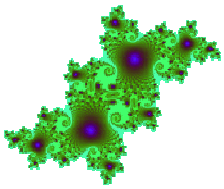
AMAÇ

Programda sıralama ile ilgili kısımları yazabileceksiniz.

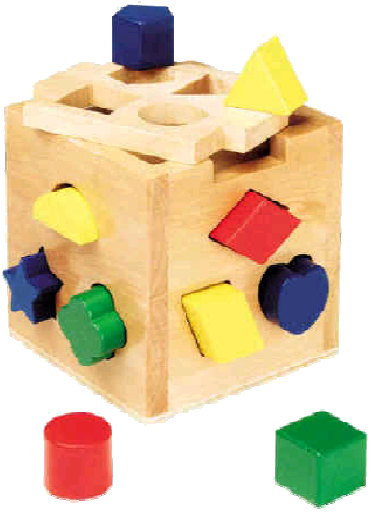
ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Kitaplığınızda karışık hâlde bulunan ansiklopedilerinizi veya dergilerinizi nasıl sıralı hâle getirirsiniz? Düzenli olarak durmasının faydaları neler olabilir?
- Bir yerden bir yere gitmek için birçok alternatifiniz olabilir. En avantajlı yolu nasıl seçersiniz? En kısa yol bulma problemi ile ilgili araştırma yapınız. Örneğin IETT sitesini ve "Google Earth" programlarını inceleyebilirsiniz.
- Doğada bulunan motifler, simetri ve fraktal hakkında araştırma yapınız. Mesela Fibonacci sayı dizisi ile ayçiçeğinin çekirdekleri aynı şekilde sıralanır.



1. SIRALAMA



Genellikle sıralama (*sorting*) işlemlerini veritabanında kullanırız. İsim bilgileri harf sırasında, telefon bilgileri alan kodlarına sıralı olarak istenebilir. Programımız bu olanakları sağlamalıdır. Çok gerekli olan bu işlemlerin algoritması karmaşık olabilir. Programınızın sıralama hızı da önemlidir. Örneğin 15 ismin sıralanması dakikalarca sürmemelidir.

Bu bölümde göreceğiniz bilgisayar bilimi çalışmaları, en kısa sürede en etkili şekilde sıralama için düzenlenmiştir.

Büyük O Gösterimi

Bir algoritmanın etkinliğini ölçmek için bilgisayar programcıları "Büyük O Gösterimi"ni tasarlamışlardır. Büyük O, bir algoritmanın yönetmesi gereken bilgi miktarının işlenme hızını ölçer.

Programcılar genellikle aynı miktardaki verinin, farklı algoritmalardaki işlem süresini bilmek isterler. Ortalama ve en kötü durum senaryosu üretirler. "Büyük O" sayesinde program için en uygun algoritma seçilir.

Mesela isimlerin sıralanacağı bir algoritmada, isimlerin sayısı programın hızını etkiler. Bu O(n) ile gösterilir. "O" sıralama büyüklüğü, "n" de nesne sayısıdır. "n" boyutundaki bir problemin çözümünde geçen adım sayısı $T(n) = 4n^2 - 2n + 2$ olarak bulunabilir.

1.1. Ekleme Sıralaması

Ekleme sıralaması (***insertion sort***) aslında bir kart oyunundaki kartların sıralanmasına benzetilebilir. Dağınık durumdaki kartlardan iki tanesini elinize alırsınız, üçüncüsünü diğerlerinin yanında uygun bir yere eklersiniz. Her kart aldığınızda diğerlerinin içinden uygun olan yere eklersiniz. Böylece kartlar sıralı hâle gelir.

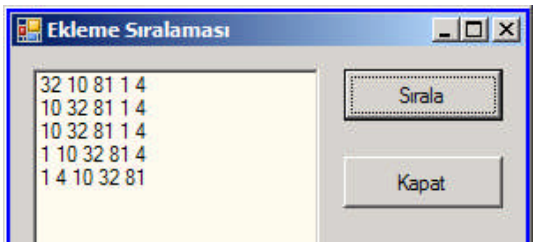
Bu yöntem ile basitçe program, karışık olan sayıları şu adımlar ile sıralar:

1. İlk iki eleman listeden alınıp, karşılaştırılır, gerekirse yerleri değiştirilir.
2. Bir sonraki eleman alınıp, önceki sıralı elemanlar içinde uygun yere eklenir.
3. İkinci adım sıralama bitene dek tekrar edilir.

Resim 1.1: Ekleme sıralaması yöntemi

Ekleme sıralamasında önce diziye* rastgele değerler yükleriz. Dizinin ikinci elemanından başlayan bir ana döngü içinde programı yazarız. Bir sonraki dizi elemanı geçici olarak bir değişkene aktarıldıktan sonra, bu değeri aktif dizi elemanı ile karşılaştırırız. Eğer "geçici değer" küçük ise başka bir döngüde, sıralı olan kısımda bu değerin yeri bulunur. Dizi sıralanana dek bu işlem devam eder.

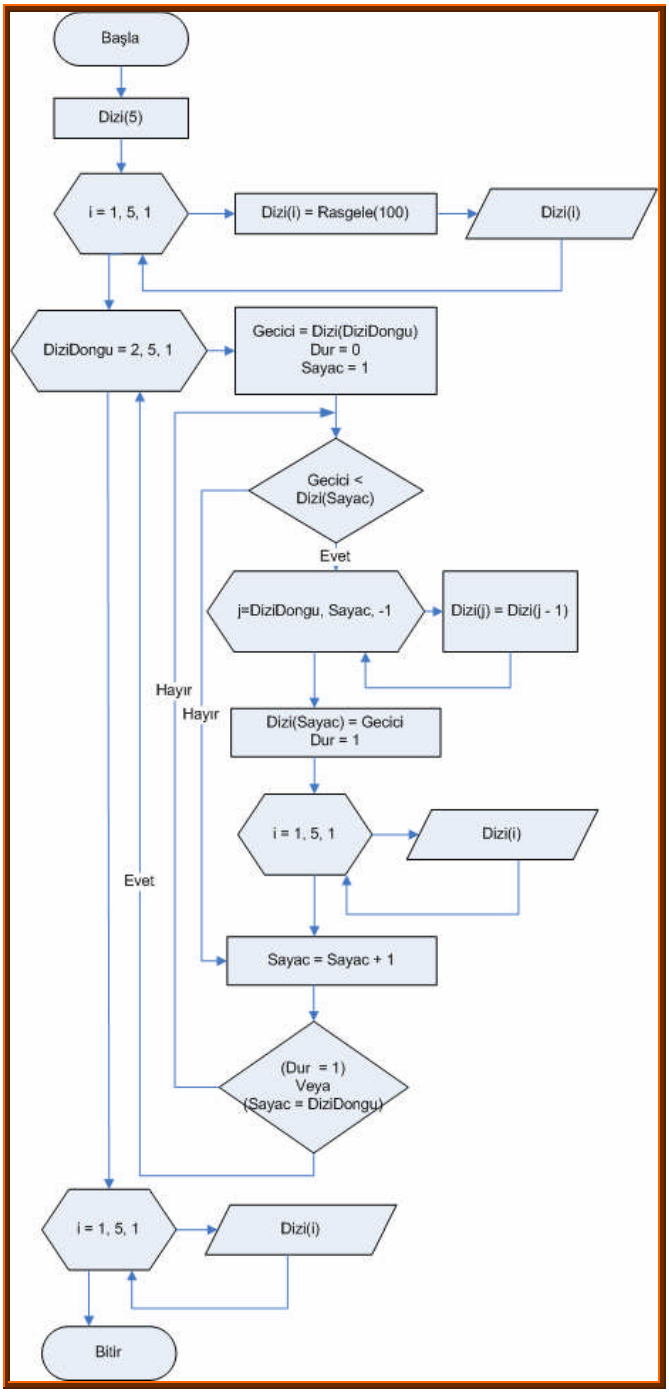
"**Ekleme Sıralaması**" programının 5 elemanlı dizi için akış şeması aşağıdaki gibidir:



Resim 1.2: "Ekleme Sıralaması"nın ekran görüntüsü

? Ekleme sıralaması örneğinin sahte kodlarını yazınız.

* Diziler, programlama dillerinde genellikle 0 veya 1 indis değeri ile başlarlar.



Resim 1.3: Ekleme sıralaması akış şeması

1.2. Balon Sıralaması

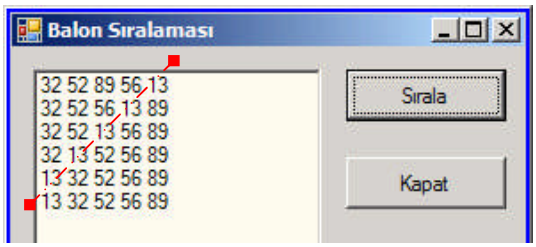
Balon sıralamasında (***bubble sort***) karışık durumdaki sayılar suyun içindeki balonlar gibi hareket ederek yerlerini bulurlar. Sayılar tekrarlı olarak kontrol edilerek yakın sayılar bir araya getirilir.

1. İlk iki eleman karşılaştırılır, gerekirse yerleri değiştirilir.
2. Listede bir sonraki elemana gidilerek, bir önceki eleman ile karşılaştırılır.
3. Liste sonuna kadar 2. adım tekrar edilir.
4. 1 ve 3. adım, tüm listenin sıralaması bitene dek tekrar edilir.

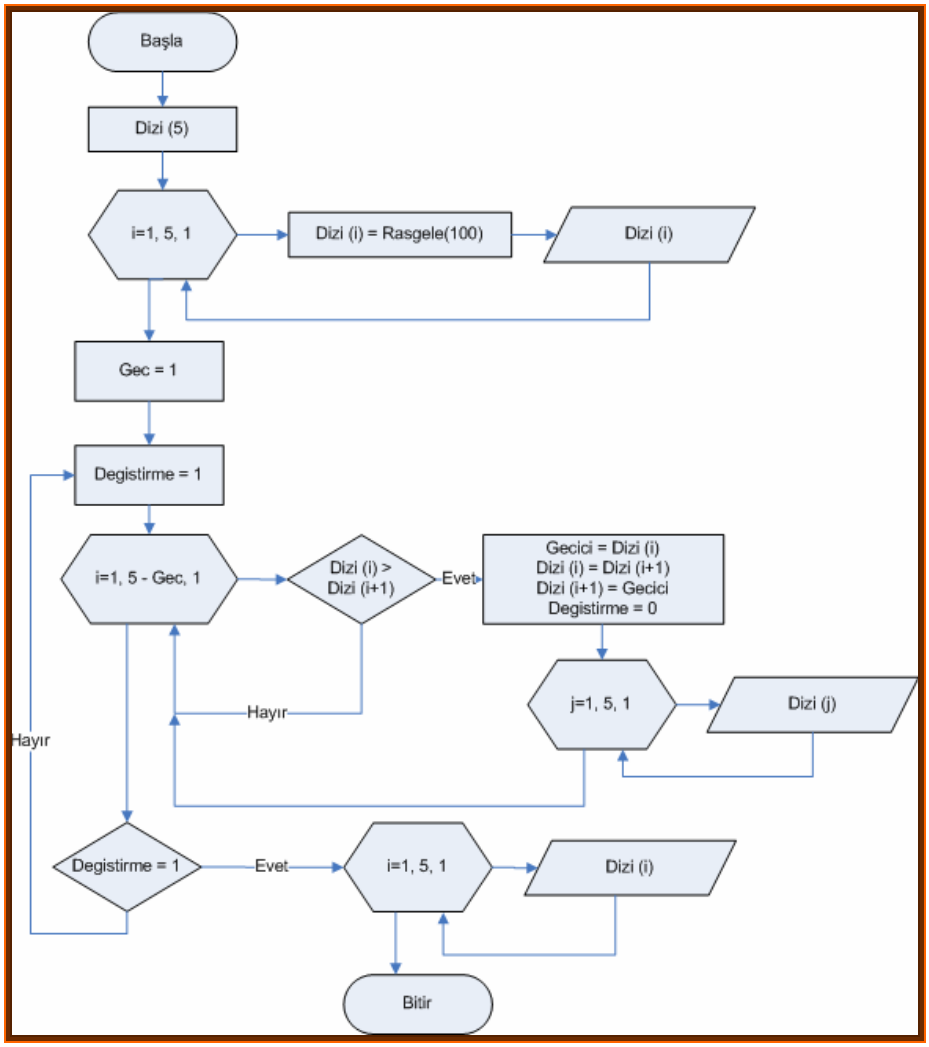
Resim 1.4: Balon sıralaması yöntemi

Önce veri listesi hazırlanır. Ön şartlı bir ana döngü içine, sınırları ilk elemandan sondan bir önceki elemana kadar olan bir döngü yapılır. Seçili elemanın değeri ile dizinin sonraki elemanının değeri karşılaştırılır. Eğer büyük ise iki dizi elemanı yer değiştirilir. Dizi sonuna kadar tarama ve yer değiştirme işlemleri devam eder. Bu döngü tekrar edilir ve değişiklik kalmamış ise ana döngü sonlandırılır.

? Balon sıralaması örneğinin sahte kodlarını yazınız.



Resim 1.5: "Balon sıralaması"nın ekran görüntüsü, 13 rakamına dikkat ediniz



Resim 1.6: Balon sıralaması akış şeması

❓ Her iki algoritmanın işlem adımlarını karşılaştırınız. Karışık listenin kaç adımda sıralandığını bulunuz.

❓ Dizilerin maksimum sınırlarını değiştirerek, kaç adımda sıralamanın bittiğini test ediniz.

1.3. Kabuk Sıralaması

Karışık durumdaki listede, en sondaki elemanı en başa getirmek zaman kaybıdır. Bu sebeple programcılar kabuk sıralaması (***shell sort***) algoritmasını geliştirmişlerdir.

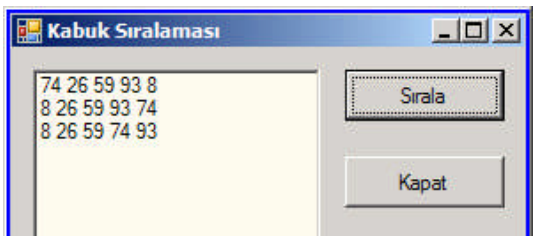
"Böl ve yönet" mantığı ile tüm dizinin sıralanması yerine, küçük parçalar halinde dizi sıralanır. Küçük listeler sıralandıktan sonra, listeler birleştirilir.

Aslında kabuk sıralaması balon ve ekleme sıralamasını hızlandırmak için geliştirilmiştir. Yani farklı bir sıralama algoritması değildir.

1. Büyük liste küçük listelere bölünür.
2. Küçük liste balon veya ekleme sıralaması ile sıralanır.
3. **Resim 1.7**'deki örnekte 15 ve 29 rakamı sıralanmasına gerek yoktur. 16 ve 4 rakamı sıralanır, 78 ise işleme girmez.
4. 3. adımda sadece 4 ve 16'nın yeri değişir, tekrar dizi küçük listelere bölünür.
5. Dizide 15, 78 ve 16 sıralanır, 4 ve 29 rakamlarının sıralanmasına gerek yoktur.
6. Liste sıralaması tamamlanana kadar 2 ve 4. adımlar tekrarlanır.

Resim 1.7: Kabuk sıralaması yöntemi

? Kabuk sıralaması sahte kodları aşağıdaki gibidir. Akış şemasını hazırlayınız.



Resim 1.8: Kabuk sıralaması ekran görüntüsü

```
Başla
          Sayısal Dizi Veriler(5)
          Sayısal i, Geçici, Dur, Geç, X, Sınır

          Yaz; "Sıralanacak veriler:"
          Döngü i = 1, 5, 1
                    Veriler(i) = Rasgele(100)
                    Yaz; Veriler(i)
          Döngü Bitti
          X = tamsayı(5 / 2)
          İken (X > 0)
                    Dur = 0
                    Sınır = 5 - X
                    İken (Dur = 0)
                              Geç = 0
                              Döngü i = 1, Sınır, 1
                                Eğer (Veriler(i) > Veriler(i + X)) İse
                                  Geçici          = Veriler(i)
                                  Veriler(i)      = Veriler(i + X)
                                  Veriler(i + X)  = Geçici
                                  Geç             = 1
                                Eğer Bitti
                              Döngü Bitti
                              Sınır = Geç - X
                              Eğer Geç = 0 İse Dur = 1
                    İken Bitti
                    X = tamsayı(X / 2)
          İken Bitti
          Yaz; "Sıralı liste:"
          Döngü i = 1, 5, 1
                    Yaz; Veriler(i)
          Döngü Bitti
Bitir
```

1.4. Hızlı Sıralama

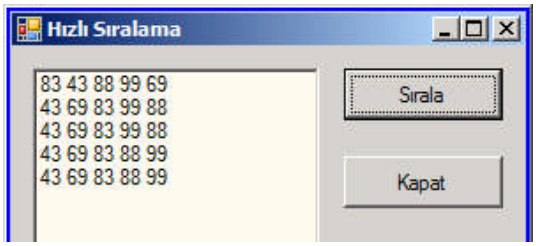
Hızlı sıralama (***quick sort***) diğer yöntemlere göre daha çok kullanılır. Bu yöntemde listenin ortasından bir eleman alınır, elemanın değerine göre sol veya sağdaki değerler yer değiştirir.

Resim 1.9: Hızlı sıralama yöntemi

Liste yarıya bölündükten sonra, her ayrılan parça tekrar yarıya bölünür. Alt parçalar kendi aralarında sıralanır. Küçük parçalar birleştirilerek tüm listenin sıralı hali oluşturulur.

1. Listenin ortasından bir eleman seçilir. Seçili elemandan büyük olan elemanlar sağa, küçük olanlar sola yer değiştirilir.
2. 1. adım listenin her yarısı için tekrar edilir.
3. Küçük listeler birleştirilir, sıralı liste elde edilir.

Kendini tekrar eden fonksiyonlara "**tekrarlamalı - *recursive***" fonksiyon denir. Basit olarak fonksiyonun kendini çağırmasıdır. "**Hızlı sıralama**"da bu yöntem kullanılıyor. Bu sebeple sıralama için alt program yapmamız gereklidir.

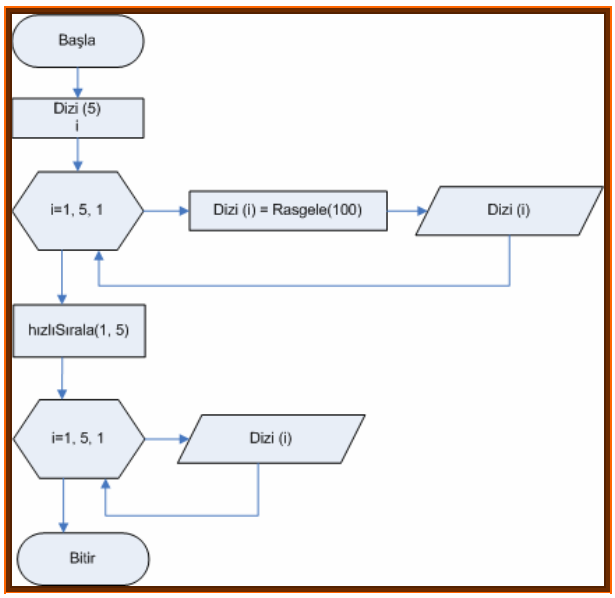


Resim 1.10: Hızlı Sıralama ekran görüntüsü

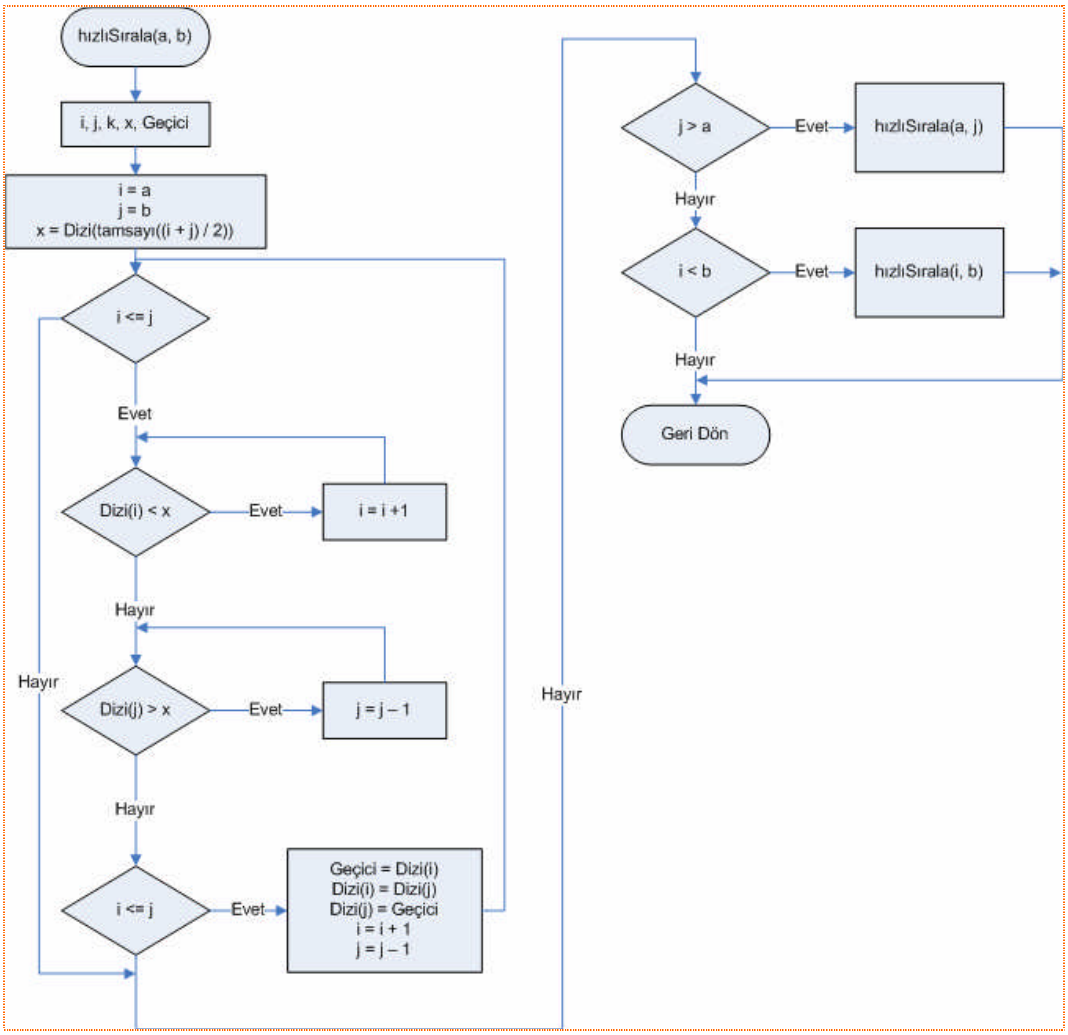
? Hızlı sıralamanın sahte kodlarını yazınız.

? Akış şemasını test ediniz. Kendiniz 5 adet rastgele rakam seçerek, bu değerleri adım adım deneyiniz.

? Sıralama yönünü "büyükten küçüğe" doğru yapmak için akış şemasında ne gibi bir değişiklik yapılmalıdır?



Resim 1.11.a: Hızlı sıralamanın ana programı



Resim 1.11b: Hızlı sıralama yönteminin akış şeması

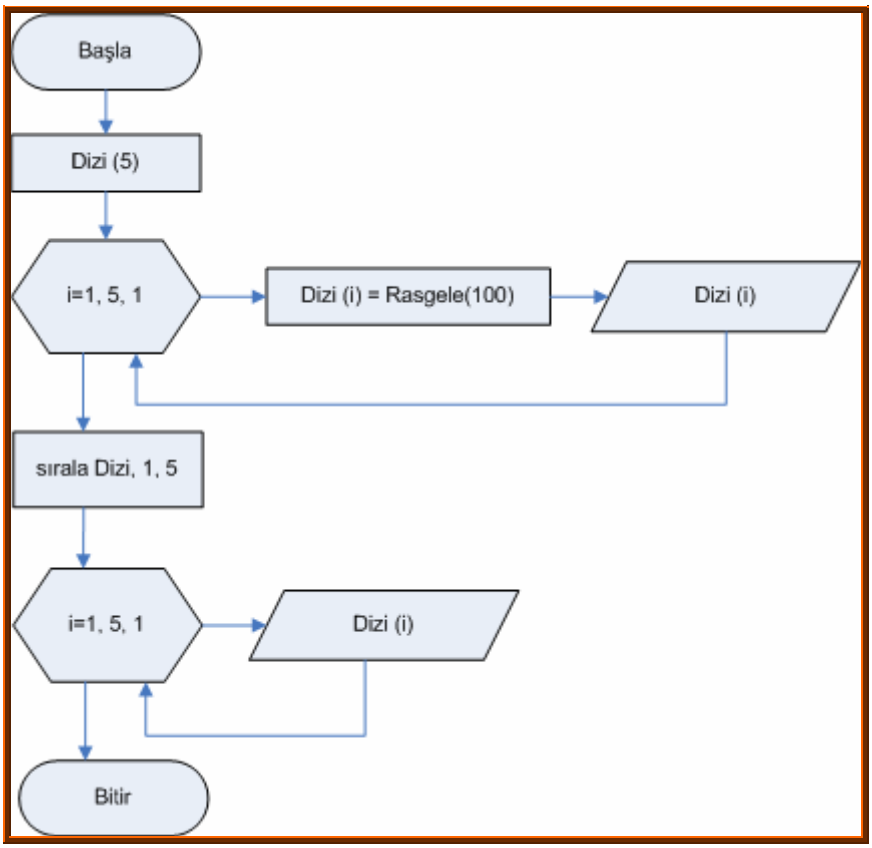
1.5. Sıralama Algoritmaları

Ekleme, balon, kabuk ve hızlı sıralama yöntemleri ile karışık listelerin değişik metotlar ile sıralanabildiğini gördünüz.

Genellikle küçük listelerde ekleme sıralaması, neredeyse sıralı olan bir listede balon sıralaması, hız gerektiren yerlerde hızlı sıralama kullanılır. Fakat kodlamak için gereken zamandan tasarruf etmek için, programcılar dilin içine "**hazır**" bulunan sıralama komutlarını tercih ederler. Örneğin, şu şekilde bir komut olabilir:

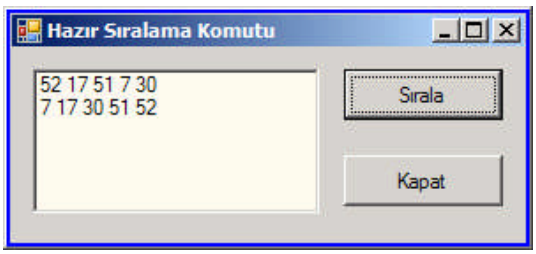
```
sırala diziAdi, ilkEleman, sonEleman
```

Dilin kendi komutunu kullanmanız tavsiye edilir, fakat komut yavaşlığa neden oluyor ise, kendi algoritmanızı oluşturunuz.



Resim 1.12: Hazır komut kullanarak sıralama işlemi

❓ Hazır sıralama yönteminin sahte kodunu yazınız.



Resim 1.13: Hazır sıralama komutu ile liste sıralamak

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. 5 adet karışık hâlde tam sayı değer seçiniz.	Aşağıdaki tablodan belli bir sütunu seçebilirsiniz.
2. Değerlerin hangi yöntem ile sıralanacağına karar veriniz.	Genellikle küçük listelerde "**balon** sıralaması", daha uzun listelerde "**eklemeli** ve **hızlı** sıralama" tercih edilir.
3. Karışık haldeki sayıları sıralayınız.	Akış şemasını çizerek, seçtiğiniz değerleri sıralayınız.
4. Ekrana sayıları listeleyiniz.	Listedeki en büyük ve en küçük değerleri istatistik amaçlı belirtebilirsiniz. Sayıların ortalama ve toplamlarını bulma işlemlerini yapabilirsiniz.
5. Programı yazarak kaç adımda sıralandığını bulunuz	Diğer algoritmalar ile işlemlerin bitmesi için gereken adım sayısı karşılaştırılabilir.

Örnek alınabilecek değerler:

	Sözcük	Resim	Sayfa	Araştırma	Örnekler
Modül 1	8664	65	54	44	14
Modül 2	8048	29	52	52	36
Modül 3	6261	29	41	22	15
Modül 4	4700	31	36	19	10
Modül 5	5608	27	39	16	12

Eğer otomobiller bilgisayarlar kadar hızlı gelişseydi, bugün bir Rolls-Royce arabayı 100 dolara alabilirdiniz ve bir litre benzinle 250.000 km yol yapabilirdiniz. Ayrıca arabanız yılda en az bir kez infilak edip içindeki herkesin ölmesine yol açardı.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. (…) Bilgisayardaki dosya ve klasörler diskte isimlerine göre sıralanmış olarak tutulur.

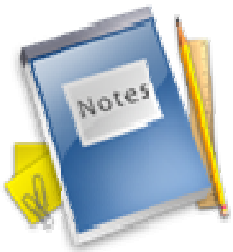
2. (…) Kısa algoritmalar karmaşık algoritmalardan daha hızlı çalışır ve hemen sonucu bulur.

3. (…) Listenin tamamını tarayan sıralama yöntemine balon sıralaması denir.

4. Aşağıdaki sıralama yöntemlerinden hangisi hemen hemen sıralı değerler için tercih edilir?
A) Eklemeli
B) Balon
C) Kabuk
D) Hızlı

5. Aşağıdaki sıralama yöntemlerinden hangisi diğer sıralama yöntemlerinden birini kullanır?
A) Eklemeli
B) Balon
C) Kabuk
D) Hızlı

6. Aşağıdaki sıralama yöntemlerinden hangisi genellikle en az adımda işlemi tamamlar?
A) Eklemeli
B) Balon
C) Kabuk
D) Hızlı



ÖĞRENME FAALİYETİ–2

AMAÇ

Programda arama ile ilgili kısımları yazabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızdaki dosya ve klasörlerin aranmasını nasıl yapıyorsunuz? Dosya içindeki bilgileri de arayabilir misiniz? Daha hızlı nasıl arama yapılabilir?
- İnternet sayfalarında arama imkânı nasıl sağlanıyor? Arama motorlarının gelişmiş arama seçeneklerini inceleyiniz.
- Elektronik posta adreslerine gereksiz mesajlar gelebilir. Bu tür mesajları bulan algoritmaları araştırınız. Mesela "Bayesian Filtreleme Algoritması" gibi…

2. ARAMA

Sıralama adımından sonra en önemli konu "**arama**" işlemidir. İsim ve adres bilgilerini saklayan bir programda, bilgilerin sıralanması ve aranması algoritmalarına ihtiyaç duyulur. Örneğin; Ankara'da yaşayan "F" harfi ile başlayan isimlerin listelenmesi istenebilir.

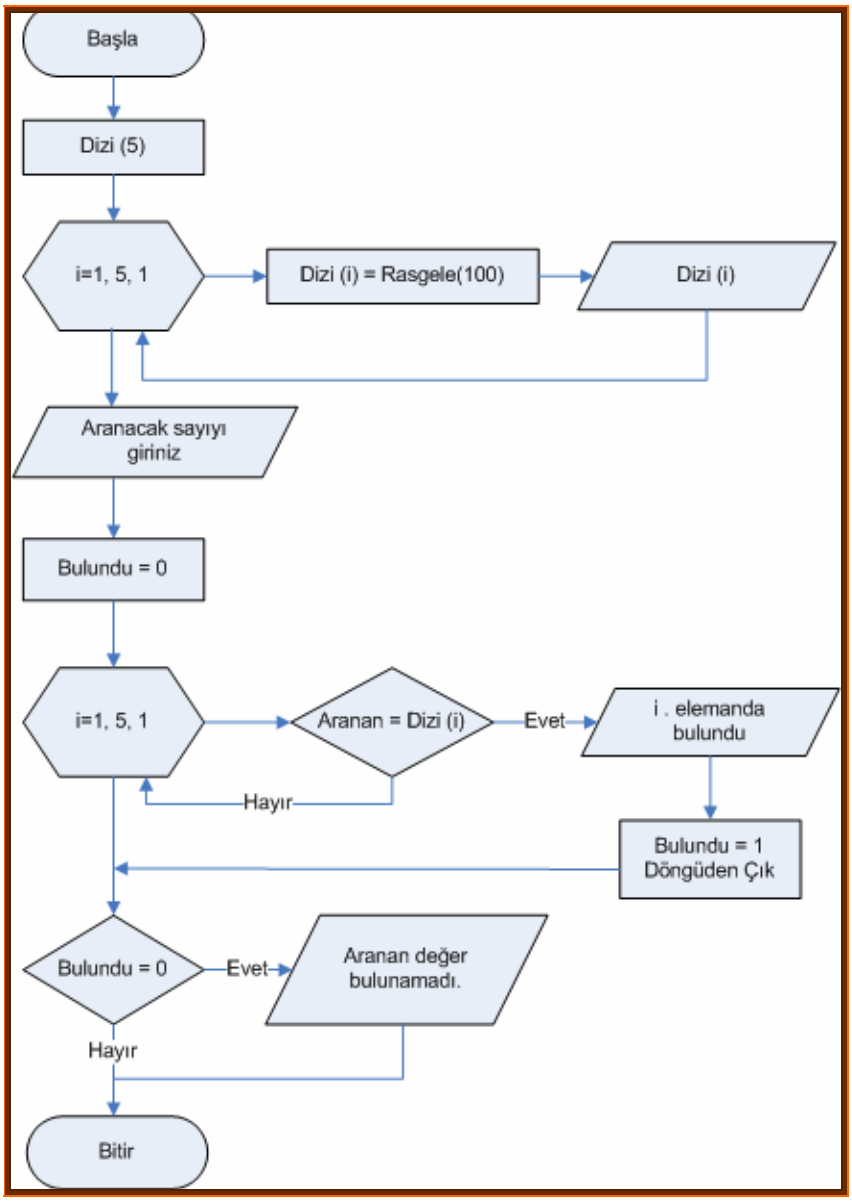
Arama işleminin hızlı olması için genellikle programlar verileri önce sıralarlar. Sıralama ve arama işlemleri, uygun algoritma seçilerek hızlı ve etkin olarak yapılır. Ayrıca sıralı listede minimum ve maksimum değer otomatik olarak bulunmuş olur. Listenin en başındaki eleman en küçük değere, liste sonundaki eleman ise en büyük değere sahiptir.

2.1. Sıralı (Ardışık) Arama

Sıralı arama (***sequential search***) listedeki tüm bilgileri tarama yöntemidir. Liste, bağlı liste veya dizi olabilir.

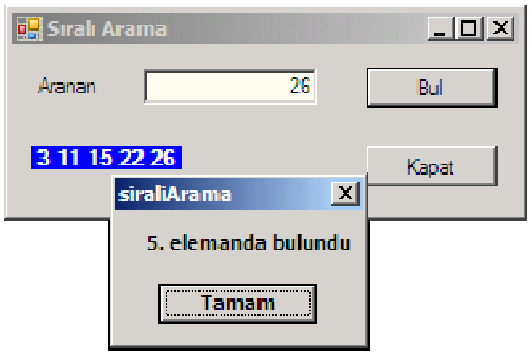
Evinizin kapı anahtarını kaybettiğinizde, sıralı arama yöntemini kullanarak anahtarı aramak istersek, apartmandaki tüm odaları tek tek arama yolu ile yapabiliriz. Eğer anahtar ilk odalarda ise hızlı bir şekilde anahtarınızı bulursunuz. Yani küçük listelerde bu arama yönteminin yavaşlığını hissetmezsiniz bile. Tüm şehri aramanız gerektiğini düşünün, arama hızı çok yavaş olurdu.

Aramayı ister başlangıçtan, isterseniz listenin sonundan başlatabilirsiniz. Aranan bilgi bulunduğunda arama işlemi sona erer.



Resim 2.1: Sıralı arama yöntemi

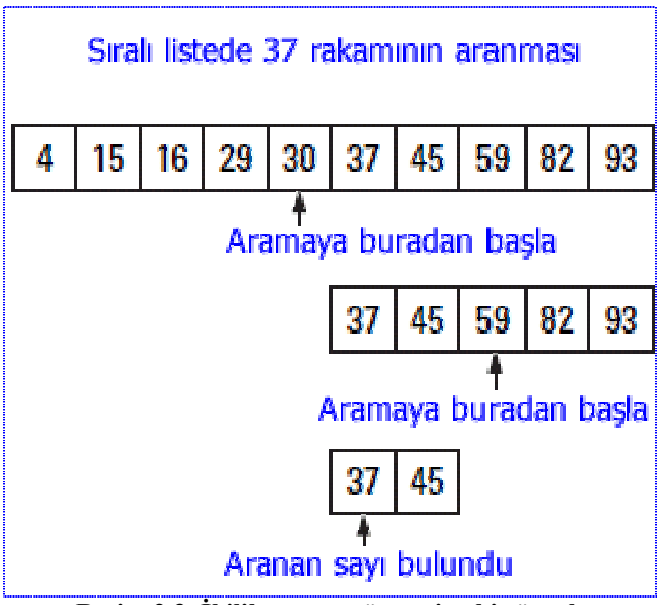
? Sıralı arama yönteminin sahte kodlarını hazırlayınız.



Resim 2.2: Sıralı arama programının ekran görüntüsü

2.2. İkilik Arama

İkilik arama[†] (***binary search***) sıralı haldeki bir listede hızlıca arama yapmamızı sağlar. Uzun liste ikiye bölünür, aranan bilgi hangi yarıda ise, o yarı içinde arama yapılır. Sayı bulunana kadar liste yarıya bölünerek arama işlemi devam eder.



Resim 2.3: İkilik arama yöntemine bir örnek

On elemanlı dizide 37 rakamını bulmak için, önce dizinin ortasındaki eleman olan 30 ile aramaya başlanır. 37 rakamı 30'dan büyük olduğu için sağ taraftaki yarıda arama yapılacaktır.

[†] İkilik arama sadece sıralı hâldeki listeler içindir.

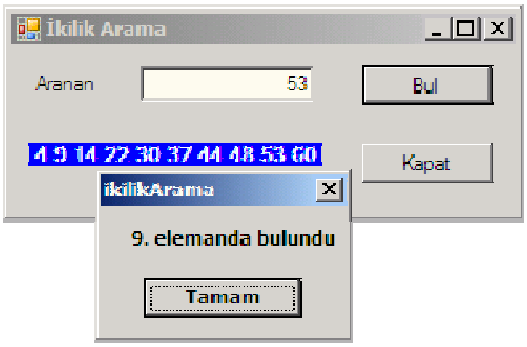
Kalan beş sayıda ortadaki 59 ile aranan sayı olan 37 karşılaştırır. Bu sefer de sol tarafta arama yapılacaktır. Elimizde iki sayı kaldı. Listedeki ilk eleman aranan sayı ile karşılaştırılır. 37 rakamı üçüncü adımda bulunmuş oldu. Sıralı arama yönteminde olsaydı altıncı adımda arama bitecekti.

❓ İkilik arama yönteminin akış şemasını çiziniz. Test değerleri seçip, akış şemanızı deneyiniz.

❓ "Yari" değişkeninin değeri bulunmasında bir açık vardır. Eğer "Sol ve Sağ" değişkenlerinin toplamı tam sayı değişkenin sınırlarını geçerse, program hata verip kapanır. Bu sorunu nasıl çözebilirisiniz?

Çözüm önerisi: (`tamsayı ()` komutu ondalıklı sayının tam sayı kısmını verir.)

```
Yari = Sol + tamsayı((Sol - Sag) / 2)
```



Resim 2.4: İkilik arama programının ekran görüntüsü

2.3. Kıyma (Hashing) Yöntemi

Kıyma veya kıyım yöntemi (***hashing search***) yeri hemen hemen bilinen değerleri bulmak için yapılmıştır. Mesela, evinizin anahtarını genellikle belli bir yere koyarız veya bir yere asarız. Böylece bulmamız kolaylaşır. Programda aramayı kolaylaştırmak için elimizdeki değerleri dizide belli yerlere atacağız.

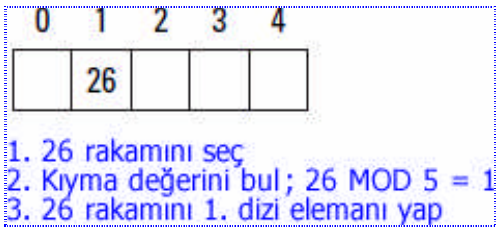
2.3.1. Kıyma Fonksiyonu

Bir veri yapısı (dizi veya bağlı liste) içine değerler için, kıyma veya kıyım değeri (***hash value***) hesaplanır. Kıyma değeri, kıyma fonksiyonu (***hash function***) yardımı ile bulunur. Kıyma fonksiyonu sayesinde, "aranan değer" tüm listede aranması yerine, belli bir yerde aranarak bulunur. Örneğin bir `arananDeger` adlı tam sayıyı dizide aradığımızı düşünelim; önce "kıyma değeri" bulunur:

```
KıymaDegeri = arananDeger % 5
//mod bulmak için başka bir yöntem:

KıymaDegeri = arananDeger – (tamsayı(arananDeger / 5) * 5)
```

Bu formül bize sayının beşe bölümünden kalan sayıyı "kıyma değeri" olarak verir. Hangi sayı saklanmış olursa olsun kıyma değerleri 0, 1, 2, 3 veya 4 olabilir. Mesela 26 rakamının 5'e bölümünden kalan değeri 1 olduğu için, 26 değerini dizinin 1. elemanına atayabiliriz.



Resim 2.5: Kıyma yöntemi ile değerleri yerleştirmek

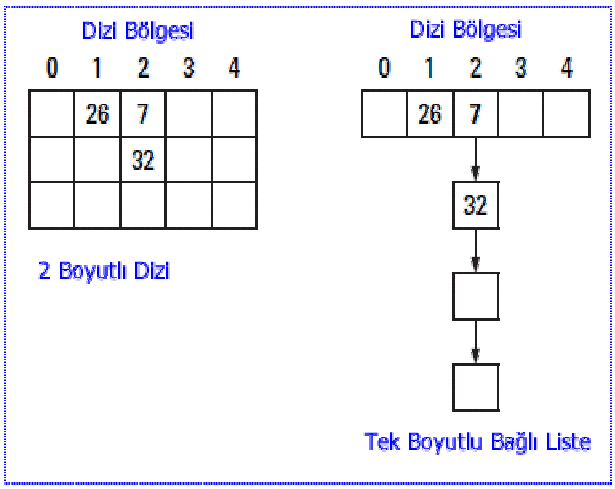
Uzun listelerde kıyma yöntemi ile arama çok hızlı bir şekilde yapılabilir.

2.3.2. Kıyma Yönteminde Çakışmalar

Kıyma fonksiyonu ile tek olan değere sahip bir sayı üretilir. Farklı liste elemanlarının aynı kıyma değeri olabilir. Mesela, 7 ve 32 değerlerinin 5 ile bölümünden kalan sayı 2'dir.

Aynı kıyma değeri birden fazla sayıda ise, buna çakışma (***collision***) denir. Çakışmaları kontrol altına almak için, aynı kıyma değerine sahip olan elemanlar bir yapıda toplanır. İki boyutlu bir dizi veya bağlı liste içine çakışanlar tutulabilir.

Dizi veya bağlı liste büyüyebilir veya hafızada fazla yer kaplamaması için kısaltılabilir.

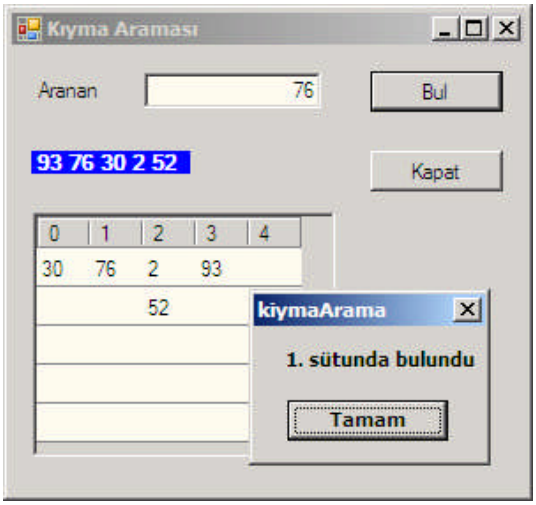


Resim 2.6: Kıyma yönteminde çakışmaları engelleme

2.3.3. Kıyma Yönteminde Arama

Değerler listeye kaydedildikten, sonra arama işlemi kıyma fonksiyonu ile rahatlıkla yapılabilir. Aynı kıyma değerine sahip elemanlar da kendi aralarında taranır.

Eğer her elemanın kendi tek (***unique***) kıyma değeri var ise, kıyma fonksiyonu sayesinde tek adımda arama işlemi tamamlanır. Birden fazla kıyma değeri olan elemanlar ise en azından küçük bir liste halinde olduğundan, arama işlemi fazla zaman kaybına neden olmaz. Bu küçük listede sıralı veya ikilik yöntemleri ile arama yapılabilir.



Resim 2.7: Kıyma yönteminin program ekran görüntüsü

? Kıyma arama yönteminde oluşan alt listelerde nasıl arama yapılabilir? Hangi arama yöntemini tercih edersiniz?

Kısa listelerde kolay kodlanabilen "**sıralı arama**", hız gerektiren daha büyük listelerde "**ikilik arama**" tercih edilir.

Çok büyük verilerde "**kıyma yöntemi**" seçilebilir, ama kodlaması zordur. Önceden değerleri veri yapısına yerleştirme gerekliliği ve alt listede farklı algoritma ile arama kullanılması gerektiği için kod yazımı uzun zaman alır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. 5 adet karışık halde tam sayı değer seçiniz.	Aşağıdaki tablodan belli bir sütunu seçebilirsiniz.
2. Aramayı kolaylaştırmak için veriyi sıralayınız.	Genellikle küçük listelerde “balon sıralaması”, daha uzun listelerde “eklemeli ve hızlı sıralama” tercih edilir.
3. Uygun algoritmayı seçerek arama programını yazınız.	Akış şemasını çizerek daha kolay program yazılabilir.
4. Arama değerleri girerek, programı deneyiniz.	Seçtiğiniz değeri listede arayabilirsiniz.

Örnek alınabilecek değerler:

	Sözcük	Resim	Sayfa	Araştırma	Örnekler
Modül 1	8664	65	54	44	14
Modül 2	8048	29	52	52	36
Modül 3	6261	29	41	22	15
Modül 4	4700	31	36	19	10
Modül 5	5608	27	39	14	12

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. Karışık bir listede sadece sayılar için arama yapabiliriz. ( )

2. Küçük listelerde sıralı arama yöntemi ile hızlıca aranan değeri buluruz. ( )

3. Çok büyük değerlere sahip olan sayıları arama işlemi daha yavaştır. ( )

4. Kıyma fonksiyonu sayesinde bir değeri ilk denemede buluruz. ( )

5. Aşağıdaki arama yöntemlerinden hangisi tüm listeyi tarayıp işlemi tamamlar?
A) Sıralı
B) İkilik
C) Kıyma
D) Hiçbiri

6. Aşağıdaki yöntemlerden hangisi listeyi ikiye bölerek arama yapar?
A) Sıralı
B) İkilik
C) Kıyma
D) Hiçbiri

ÖĞRENME FAALİYETİ–3

AMAÇ

Programın derlenmesini ve kod yazımını daha iyi hâle getirebileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızın yazılım ve donanım olarak daha iyi çalışması için neler yapıyorsunuz? Mesela disk birleştirici ile ayda bir disklerin birleştirilmesi gibi…
- Bilgisayarınızda kurulu bir program artık çalışmamaya başlarsa veya işletim sisteminde problemler meydana gelmiş ise ne gibi önlemler alabilir, tamir etmek için neler yaparsınız?

3. KOD İYİLEŞTİRME

Program çalışır hâle geldikten, hatalar elinden geldiğince giderildikten sonraki istenen yazılımın sürümünün oluşturulması veya kodların iyileştirilmesidir. **İyileştirme** ile anlatılmak istenen şunlardır:

1. Programı daha da hızlandırmak
2. Programın kapladığı disk alanını azaltmak
3. Program için gereken bellek miktarını azaltmak

Programın ilk sürümü 1.0 olarak piyasaya sürüldükten belli bir süre sonra, yazılım firmaları küçük güncellemeler yaparak 1.01 gibi yeni sürümler oluştururlar. Genellikle sonradan bulunan böcekler giderilmiş olur, çok büyük yenilik yapılmaz. Yepyeni yapılan değişikliklerle 2.0 gibi daha büyük sürüm numarası ilerlemesi yapılır.

3.1. Doğru Veri Yapısını Seçmek

Her programın veri saklama ihtiyacı vardır. Bunun için doğru yöntemi seçmelisiniz. Bir **dizi** yapmak çok kolay gözükebilir, ama dizinin eleman sayısını bilmeniz gereklidir. Küçük bir dizi yaptıysanız, programınız daha fazla bilgi saklamak istediğinde çökebilir; büyük bir dizi yaptıysanız, ana bellekte gereksiz alan kaplayan içi boş değişkenler oluşur. Gereğinden fazla bellekte yer kaplayan diziden uzak durulmalıdır.

Seçtiğiniz veri yapısı kullandığınız sıralama ve arama algoritmasını etkiler. Bir dizi yerine bağlı liste kullanırsanız, daha etkin ve hızlı olarak veriler içinde yer değiştirme işlemi yapabilirsiniz.

3.2. Doğru Algoritmayı Seçmek

Algoritma ile bir problemin nasıl aşılacağı ayrıntılı olarak anlatılır. Mesela arkadaşınıza evinizi tarif etmeniz gerekirse, kısa olsun diye ara sokaklardan veya kolay olsun diye ana yoldan gitmeyi anlatabilirsiniz. Ara sokakların anlatılması zordur, ana yolun tarifi de kolay ama yolu uzatıyor olabilir.

50.000 kişinin listelendiği bir veri yapısında "**hızlı** sıralama" yerine, "**balon** sıralaması" yapmak çok uzun zaman alır. Sıralama yapılan listede arama yapmak için, "**sıralı** arama" yöntemi çok uzun zamanda sonuçlanır, "**ikilik** arama" yöntemi seçilerek daha hızlı tamamlanabilir.

Bir oyundaki en yüksek 10 rekor bilgisi saklanıyor diyelim, ilk kez oyun oynandığında 10 adet rekor bilgisi boştur. Her oyun oynandığında rekora yeni değerler eklenir. Burada en uygun sıralama yöntemi "**ekleme** sıralaması" olabilir. Yeni yüksek rekor yapıldıkça listede uygun yere bilgi eklenir. Eğer "**balon** sıralaması" ile rekor bilgilerini sıralamak isterseniz, her oyun sonunda bilgiler defalarca taranarak sıralanır.

Yazdığınız programda, doğru algoritmayı seçerek programın daha hızlı çalışmasını sağlayabilirsiniz.

3.3. Kaynak Kodu İyileştirmek

Doğru veri yapısını ve algoritmayı seçseniz bile kodu iyileştirme (***code optimizing***) gerekebilir. Programınızın belli kısımlarını tekrar yazarak daha hızlandırabilir ve daha az bellek harcar hale getirebilirsiniz.

Öneriler:

- "Eğer" komutunda birden fazla şartı "Ve / Veya" işleçleri ile bağlayınız.
  Örnek:
  ```
  Eğer (şart1) Ve (şart2) İse
     Komutlar
  Eğer Bitti
  ```
  Burada birinci şart eğer Hayır (False) ise ikinci veya diğer şartlara bakılmaz. Bu sayede program fazladan işlem yapmaz. Zaman kazanırız ve program biraz daha hızlı çalışır.

- İç içe "Eğer" kullanırken Evet (True) olması muhtemel şartı önce yazınız. "Durum" komutunda da buna dikkat ediniz. Böylece diğer şartları boşuna taramak gerekmez.
  Örnek:
  ```
  Eğer şart1 İse
   Komutlar
  Değilse Eğer şart2 İse
   Komutlar
  Eğer Bitti
  ```

Eğer birinci şart doğru ise ikinci şarta bakılmaz. Eğer çoğu zaman birinci şart Hayır (False) ikinci şart Evet (True) oluyor ise şartların yerlerini değiştirerek, zaman ve hız kazanabilirsiniz.

- “Döngü” komutu ile gereksiz fazladan döngü yapmayınız. Döngüler programda çok zaman yitirilmesine sebep olabilir. Döngü belli sayıda iç kısmındaki komutları çalıştırır. Mesela dizide aranan değer daha ilk eleman ise, diğer elemanlar taranmaya devam edilmemelidir. Zaman kaybı olmaması için gerekli yere “Döngüden Çık” komutu yerleştirerek döngüden hedefe ulaşılınca çıkılır.

```
Bulundu = 0
Döngü J = 1, 300, 1
          Eğer Dizi(J) = Aranan İse
                    Bulundu = 1
                    Döngüden Çık //döngüden çıkılacak
          Eğer Bitti
Döngü Bitti
Eğer bulundu = 0 İse
          Yaz; "Aranan değer bulunamadı!"
Değilse
          Yaz; "Aranan değer bulundu: " & J
Eğer Bitti
```

- Döngü içine gereksiz kod yazmayınız. Döngü ile alakası olmayan satırları döngü dışına alınız.

```
Döngü J = 1, 5000, 1
          I = 0
          Eğer Dizi(J) = 55 İse
            Yaz; Dizi(J)
          Eğer Bitti
Döngü Bitti
```

Arama yapılırken fazladan yazılan “`I = 0`” satırı 5.000 kez çalıştırılacaktır. Bu gereksiz satır, az da olsa döngünün yavaşlamasına sebep olacak, sonuç olarak programınızı yavaşlatacaktır.
İç içe döngü yaparken içteki döngüye gereksiz kod yazmayınız. İç döngü yavaşlar ise, dış döngü de yavaşlar.

- Bellek tasarrufu için değişkenlerin veri türünü doğru seçiniz. Tam sayı bir değişken içine küçük değer aktarılarak kullanılıyor ise “Short – Kısa”, büyük değerler aktarılarak kullanılıyor ise “Long – Uzun” seçebilirsiniz. Kısa tam sayının alabileceği değer aralığı yaklaşık 64.000 iken, uzun tam sayının değer aralığı yaklaşık 4 milyardır. Short türünde bir tam sayı 2 Byte, Long ise 8 Byte bellek alanı kaplar.

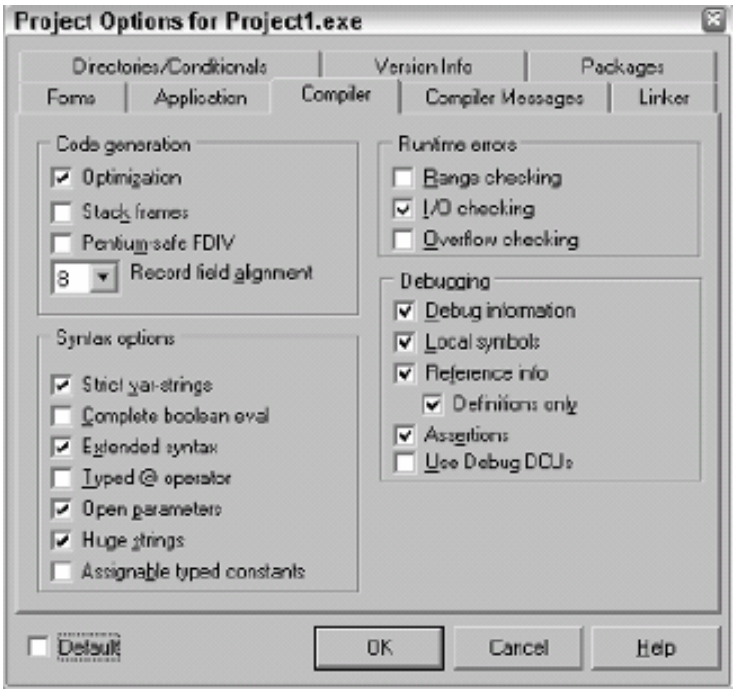
- Mümkün olduğunca programlama dilinin kendi komutları ile hesaplama yapınız. Genellikle bu komutlar daha hızlı çalıştırılırlar.

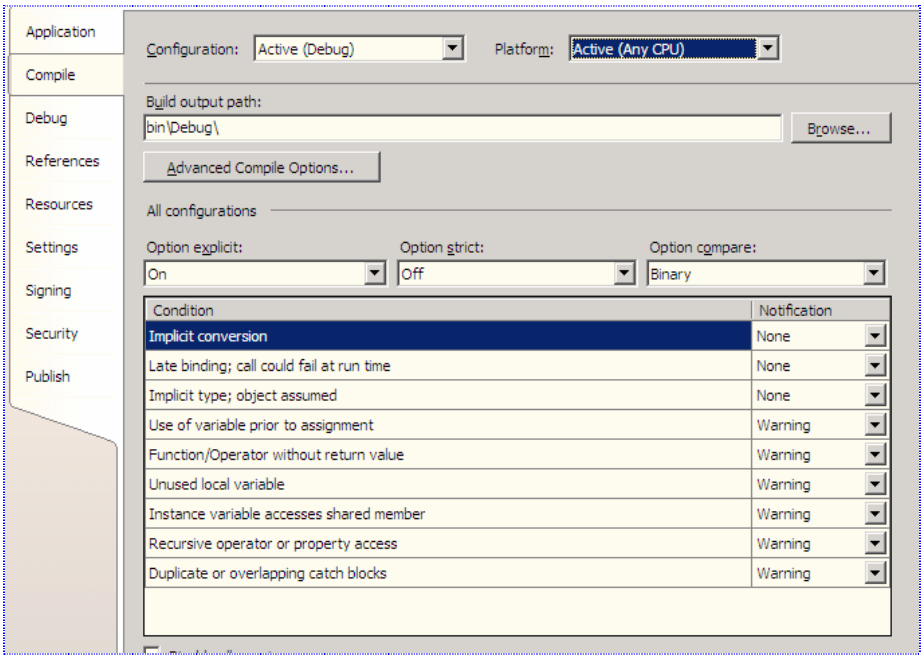
```
Sayi = Sayi + 1       //komutu yerine
Sayi++;               //C dilinde
Inc(Sayi);            //Pascal dilinde
```

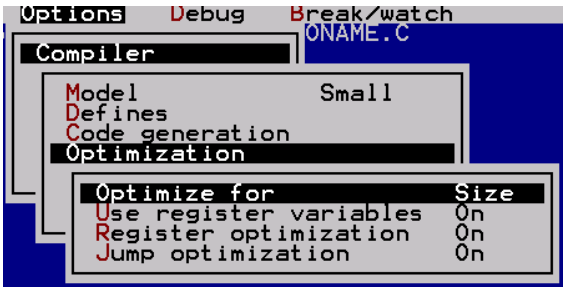
Dilin kendi komutlarını kullanmak kodun anlaşılır olmasını engeller. Diğer programcılar kodu anlayamayabilirler.

- Daha hızlı bir programlama diline geçiniz. En hızlı programlama dili makine dilidir. Sonra C ve C++ dilleri gelir. Birçok programcı basit ve yavaş olan Visual Basic'te prototip olarak programı hazırlar, daha sonra asıl programlama diline kodları taşırlar. Aslında bu zahmetli ve riskli bir işlemdir. Yeni kodların çalışacağı garanti değildir.
  Öyleyse ikinci bir yol, tüm programı yazmak yerine görüntü olarak basit bir dilde hazırlamak, asıl dile sonra geçmek olabilir.
  Birden fazla dil kullanılarak program yazılıyor ise, her dilin kendi avantajı kullanılabilir. Birbiri ile çalışabilen dillerde program yazmak, farklı programlama dilleri bilen programcıların **takım** halinde bir proje üzerinde çalışmalarını sağlar.

- Derleyicinizi daha iyi hâle getiriniz. Daha hızlı bir dil yerine daha performanslı bir derleyici seçilebilir.
  Derleyicinin ayarlarını inceleyerek, daha iyi hâle getirebilirsiniz. Ayarları değiştirirken dikkatli olunuz, programı hızlandırmak isterken, hata ve çökmelere karşı daha zayıf hâle getirebilirsiniz.







Resim 3.1: Delphi, VB2005 ve C derleyici seçenekleri

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Programın bitiminde kodları iyileştiriniz.	Öneriler kısmındaki maddelere göre bir örnek programı iyileştiriniz.
2. Kod satırlarını daha okunaklı hale getiriniz.	Yaptığınız programı diğer programcıların rahat anlaması için okunaklı hâle getirmeniz iyi bir alışkanlıktır.
3. Program için daha hızlı bir dile kodları dönüştürünüz.	QBasic gibi bir dilde yazılan programı daha performanslı olan C gibi bir dile çeviriniz.
4. Derleyici seçeneklerini iyileştiriniz.	Programlama dilinin derleme seçeneklerini inceleyiniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. Programların sürüm numaraları her zaman 1.0 ile başlar. ( )

2. Programların sürüm numaraları sadece çok büyük değişikliklerde ilerler. ( )

3. Basit mantık kullanılarak yapılan algoritmalar, programın performansını artırır. ( )

4. Derleyici ayarları değiştirilerek programın hiç hata vermeden çalışması sağlanabilir, fakat programın çökme ihtimali daha da artar. ( )

5. Aşağıdakilerden hangisi programın çalışmasını hiç etkilemez?
A) Programdaki gereksiz açıklama satırlarını silmek
B) Programda iç içe döngüleri çok kullanmak
C) Programda "Eğer" yerine "Durum" komutunu kullanmak
D) Programdaki değişkenlerin türünü doğru seçmek

6. Aşağıdakilerden hangisi kodu iyileştirme ile ilgili değildir?
A) Programı daha hızlandırmak
B) Programın kapladığı disk alanını azaltmak
C) Programın yardım dosyasını hazırlamak
D) Program için gereken bellek miktarını azaltmak

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 9 puan arasında olacak şeklinde değerlendirecektir.

DEĞERLENDİRME KRİTERLERİ	Puan
5 adet karışık hâlde tam sayı değer seçme	
Değerlerin hangi yöntem ile sıralanacağına karar verme	
Karışık hâldeki sayıları sıralama	
Programı yazarak kaç adımda sıraladığını bulma	
Aramayı kolaylaştırmak için veriyi sıralama	
Uygun algoritmayı seçerek arama programını yazma	
Arama değerleri girerek, programı deneme	
Programın bitiminde kodları iyileştirme	
Kod satırlarını daha okunaklı hâle getirme	
Program için daha hızlı bir dile kodları dönüştürme	
Derleyici seçeneklerini iyileştirme	
Toplam (en fazla 99 puan olabilir)	

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır, öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	Y
2	Y
3	D
4	B
5	C
6	D

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	Y
2	D
3	Y
4	Y
5	A
6	B

ÖĞRENME FAALİYETİ-3 CEVAP ANAHTARI

1	Y
2	Y
3	Y
4	D
5	A
6	C

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

- "0,1,2,3,4,5,6,7,8,9,A,B,C,D..." diye sayıyorsanız...
- Rüyalarınız 256 renkse...
- Uyumaya çalışırken sleep(8 * 3600) diye düşünüyorsanız...
- Asansöre bindiğinizde gitmek istediğiniz kata ait düğmeyi çift tıklıyorsanız...

... biraz dikkatli olsanız iyi olur.

SÖZLÜK

İsim	Okunuş	Anlam
sign	sayn	işaret, signature – imza
simulate	simyuleyt	benzetmek (simulation - benzetim)
slot	slot	yuva, kart takılan yarıklar
specify	spesifay	tanımlamak, belirtmek
speech	spiiç	konuşma
sprite	sprayt	grafik programcılığında kullanılan hareketli nesneler
SQL	es kyu el	Structured Query Language – Yapısal Sorgulama Dili
stack	stek	yığın, geçici bellek alanı
status	steytıs	durum; statement – deyim, komut
structure	strakçır	yapı
subdirectory	sabdayrektöri	alt dizin, klasör
substitute	sabstityut	başkasının yerine koymak
success	sıkses	başarı
supervisor	supıvayzır	yetkili
support	sıport	destek
surface	söfıs	yüzey, görünüş
suspend	sıspend	askıya almak
switch	sviç	anahtar, yer değiştirmek
terminal	törminıl	son, uçbirim
thread	tred	kanal
toggle	togl	iki şeyin birbirleri ile yer değiştirmeleri (swap)
track	trek	iz, fiziksel disk dairesi
true	tru	mantıksal doğru
truetype	trutayp	boyutu değiştirilebilir yazı tipi
try	tray	denemek
tutorial	tyutırıl	öğretici
unit	yunit	kod yazılan alan
utility	yutiliti	yardımcı programlar
variable	verayıbl	değişken
virus	vayrıs	virüs
volume	volyum	sürücü
warning	worning	uyarı
watch	voç	çalışma anında değişkenleri seyretmek

Visual Basic dilinde ekleme sıralaması örneği

```
Private Sub Button1_Click
        'Kurulum kısmı
        Dim Maksimum As Integer = 5
        Dim Dizi(Maksimum) As Integer
        Dim i, j, DiziDongusu, Gecici, Dur, Dur2, Sayac As Integer
        Dim metin As String
        'Rasgele komutu için Randomize() ekledik
        Randomize()
        ListBox1.Items.Clear()
        'İlk değerler atanıyor
        metin = ""
        For i = 1 To Maksimum
            Dizi(i) = Rnd(1) * 99 + 1
            metin = metin & Dizi(i) & " "
        Next i
        ListBox1.Items.Add(metin)
        'Sıralama döngüsü açılıyor
        For DiziDongusu = 2 To Maksimum
            Gecici = Dizi(DiziDongusu)
            Dur = 0 : Sayac = 1 : Dur2 = 0
            While (Dur2 = 0)
                If Gecici < Dizi(Sayac) Then
                    For j = DiziDongusu To Sayac Step -1
                        Dizi(j) = Dizi(j - 1)
                    Next j
                    Dizi(Sayac) = Gecici
                    Dur = 1
                End If
                Sayac = Sayac + 1
                If (Dur = 1) Or (Sayac = DiziDongusu) Then Dur2 = 1
            End While
            'Ekrana ara değişiklikler yansıtılıyor
            metin = ""
            For i = 1 To Maksimum
                metin = metin & Dizi(i) & " "
            Next i
            ListBox1.Items.Add(metin)
        Next DiziDongusu
 End Sub
```

Visual Basic dilinde balon sıralaması örneği

```
'Kurulum kısmı
Dim Maksimum As Integer = 5
Dim Dizi(Maksimum) As Integer
Dim i, j, Gecici, Dur, Gec, Degistirme As Integer
Dim metin As String
'Rasgele komutu için Randomize() ekledik
Randomize()
ListBox1.Items.Clear()
'İlk değerler atanıyor
metin = ""
For i = 1 To Maksimum
          Dizi(i) = Rnd(1) * 99 + 1
          metin = metin & Dizi(i) & " "
Next i
ListBox1.Items.Add(metin)
'Sıralama döngüsü açılıyor
Gec = 1
Dur = 0
While (Dur = 0)
            Degistirme = 1
            For i = 1 To (Maksimum - Gec)
                If Dizi(i) > Dizi(i + 1) Then
                    Gecici = Dizi(i)
                    Dizi(i) = Dizi(i + 1)
                    Dizi(i + 1) = Gecici
                    Degistirme = 0
                End If
            Next i
            metin = ""
            For j = 1 To Maksimum
                metin = metin & Dizi(j) & " "
            Next j
            ListBox1.Items.Add(metin)
            If Degistirme = 1 Then Dur = 1
End While
```

Visual Basic dilinde kabuk sıralaması örneği

```
'Kurulum kısmı
Dim Maksimum As Integer = 5
Dim Dizi(Maksimum) As Integer
Dim i, Gecici, Dur, Gec, X, Sinir As Integer
Dim metin As String
'Rasgele komutu için Randomize() ekledik
Randomize()
ListBox1.Items.Clear()
'İlk değerler atanıyor
metin = ""
For i = 1 To Maksimum
          Dizi(i) = Rnd(1) * 99 + 1
          metin = metin & Dizi(i) & " "
Next i
ListBox1.Items.Add(metin)
'Sıralama döngüsü açılıyor
X = Int(Maksimum / 2)
While X > 0
            Dur = 0
            Sinir = Maksimum - X
            While (Dur = 0)
                Gec = 0
                For i = 1 To Sinir
                    If Dizi(i) > Dizi(i + X) Then
                        Gecici  = Dizi(i)
                        Dizi(i) = Dizi(i + X)
                        Dizi(i + X) = Gecici
                        Gec = i
                    End If
                Next i
                Sinir = Gec - X
                If Gec = 0 Then Dur = 1
            End While
            metin = ""
            For i = 1 To Maksimum
                metin = metin & Dizi(i) & " "
            Next i
            ListBox1.Items.Add(metin)
            X = Int(X / 2)
End While
```

Visual Basic dilinde hızlı sıralama örneği

```
Public Class Form1
    'Kurulum kısmı (global değişkenler)
    Dim Maksimum As Integer = 5
    Dim Dizi(Maksimum) As Integer

Sub hizliSirala(ByVal Baslangic, ByVal Bitis)
        'Alt programımız
        Dim metin As String
        Dim i, J, k, X, Gecici As Integer
        i = Baslangic : J = Bitis
        X = Dizi(Int((i + J) / 2))
        While i <= J
            While Dizi(i) < X
                i = i + 1
            End While
            While Dizi(J) > X
                J = J - 1
            End While
            If i <= J Then
                Gecici  = Dizi(i): Dizi(i) = Dizi(J): Dizi(J) = Gecici
                i = i + 1
                J = J - 1
            End If
        End While
        metin = ""
        For k = 1 To Maksimum
            metin = metin & Dizi(k) & " "
        Next k
        ListBox1.Items.Add(metin)
        'Alt program kendini çağırıyor
        If J > Baslangic Then hizliSirala(Baslangic, J)
        If i < Bitis Then hizliSirala(i, Bitis)
End Sub

Private Sub Button1_Click
        'Sırala düğmesi kodları (ana program)
        Dim metin As String
        Dim i As Integer

        Randomize()
        ListBox1.Items.Clear()
        'İlk değerler atanıyor
        metin = ""
        For i = 1 To Maksimum
            Dizi(i) = Rnd(1) * 99 + 1
            metin = metin & Dizi(i) & " "
        Next i
        ListBox1.Items.Add(metin)
        'Sıralama alt programı çağrılıyor
        hizliSirala(1, Maksimum)
        metin = ""
        For i = 1 To Maksimum
            metin = metin & Dizi(i) & " "
        Next i
        ListBox1.Items.Add(metin)
End Sub
```

Visual Basic dilinde hazır sıralama komutu örneği

```
Dim Maksimum As Integer = 5
Dim Dizi(Maksimum) As Integer
Dim i As Integer
Dim metin As String

Randomize()
ListBox1.Items.Clear()
'İlk değerler atanıyor
metin = ""
For i = 1 To Maksimum
          Dizi(i) = Rnd(1) * 99 + 1
          metin = metin & Dizi(i) & " "
Next i
ListBox1.Items.Add(metin)
'hazır komut kullanıyoruz
Array.Sort(Dizi, 1, Maksimum)
metin = ""
For i = 1 To Maksimum
          metin = metin & Dizi(i) & " "
Next i
ListBox1.Items.Add(metin)
```

Visual Basic dilinde sıralı arama örneği

```
Dim Maksimum As Integer = 4
Dim Dizi(), I, Bulundu, Aranan As Integer

Private Sub Form1_Load
        'Programın açılış anında çalışan komutlar

        Dim Metin As String
        ReDim Dizi(Maksimum)

        Randomize()
        Dizi(0) = Int(Rnd(1) * 10) + 1
        Metin = Dizi(0)
        TextBox1.Text = Metin
        For I = 1 To Maksimum
            Dizi(I) = Dizi(I - 1) + Int(Rnd(1) * 10) + 1
            Metin = Metin & " " & Dizi(I)
        Next I
        Label2.Text = Metin
End Sub

Private Sub Button1_Click
        'Bul düğmesinin komutları

        Aranan = TextBox1.Text
        Bulundu = 0
        For I = 0 To Maksimum
            If Dizi(I) = Aranan Then
                Bulundu = 1
                MsgBox(I + 1 & ". elemanda bulundu")
                Exit For
            End If
        Next I
        If Bulundu = 0 Then MsgBox("Aranan değer bulunamadı.")
End Sub
```

Visual Basic dilinde ikilik arama örneği

```
Dim Maksimum As Integer = 9
Dim Dizi(), I, Aranan As Integer

Private Sub Form1_Load
        'Programın açılış anında çalışan komutlar
        Dim Metin As String
        ReDim Dizi(Maksimum)

        Randomize()
        Dizi(0) = Int(Rnd(1) * 10) + 1
        Metin = Dizi(0)
        For I = 1 To Maksimum
            Dizi(I) = Dizi(I - 1) + Int(Rnd(1) * 10) + 1
            Metin = Metin & " " & Dizi(I)
        Next I
        Label2.Text = Metin
        TextBox1.Text = Dizi(Int(Rnd(0) * Maksimum))
End Sub

Private Sub Button1_Click
        'Bul düğmesinin komutları
        Dim Dur, Sol, Sag, Yari As Integer
        Aranan = TextBox1.Text
        Sol = 0
        Sag = Maksimum
        Dur = 0
        While Dur = 0
            Yari = Int((Sol + Sag) / 2)
            If Aranan < Dizi(Yari) Then
                Sag = Yari - 1
            Else
                Sol = Yari + 1
            End If
            If (Aranan = Dizi(Yari) Or Sol > Sag) Then
                Dur = 1
                Exit While
            End If
        End While
        If (Aranan = Dizi(Yari)) Then
            MsgBox(Yari + 1 & ". elemanda bulundu")
        Else
            MsgBox("Aranan değer bulunamadı.")
        End If
End Sub
```

Visual Basic dilinde kıyma arama örneği

```
Dim Maksimum As Integer = 5
Dim Dizi(,) As Integer
Dim i, j, Aranan As Integer

Private Sub Form1_Load
        ReDim Dizi(Maksimum, Maksimum)
        Dim kiymaDegeri, Sayac, Saklanan, Dur As Integer
        Dim metin As String

        Randomize()
        For i = 1 To Maksimum       'dikey değerler
            For j = 1 To Maksimum   'yatay değerler
                Dizi(i, j) = 0
            Next j
        Next i
        DataGridView1.Rows.Add(5)
        Sayac = 1 : metin = ""
        For J = 1 To Maksimum
            Dur = 0
            Saklanan = Int(Rnd(1) * 100) + 1
            kiymaDegeri = Saklanan - (Int(Saklanan / 5) * 5)
            While Dur <> 1
                If Dizi(Sayac, kiymaDegeri) = 0 Then
                    Dizi(Sayac, kiymaDegeri) = Saklanan
                    Dur = 1
                Else
                    Sayac += 1
                End If
            DataGridView1.Item(kiymaDegeri, Sayac - 1).Value = Saklanan
            End While
            metin &= Saklanan & Space$(1)
        Next J
        Label2.Text = metin
End Sub

Private Sub Button1_Click
        Dim kiymaDegeri, Aranan As Integer
        Aranan = TextBox1.Text
        kiymaDegeri = Aranan - (Int(Aranan / 5) * 5) + 1
        If Dizi(1, kiymaDegeri) = 0 Then
            MsgBox("Aranan değer bulunamadı.")
        Else
            MsgBox(kiymaDegeri & ". sütunda bulundu")
        End If
End Sub
```

ÖNERİLEN KAYNAKLAR

- blogs.borland.com/corbindunn/archive/2004/09/29/1483.aspx
- en.wikipedia.org/wiki/Fractal
- fraktaller.tripod.com
- googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html
- main.linuxfocus.org/Turkce/March2003/article273.shtml
- ostermiller.org/ti82
- ozguryazilim.org
- tr.wikipedia.org/wiki/Big-O
- tr.wikipedia.org/wiki/Google_Earth
- tr.wikipedia.org/wiki/Seyyar_satıcı_problemi
- www.algosort.com
- www.automatedqa.com/products/aqtime
- www.ba.infn.it/~zito/plaw.html
- www.belgeler.org/howto/acik-kod-yazilimcisi.html
- www.codeproject.com
- www.cs.ubc.ca/~harrison/Java
- www.developerfusion.co.uk/show/3824
- www.devx.com/vb2themax/Article/19900?type=kbArticle&trk=MSCP
- www.e-optimization.com
- www.iett.gov.tr
- www.ileriseviye.org/Makale/lisperati-tr
- www.ileriseviye.org/arasayfa.php?inode=kadinlar.html
- www.inner-smile.com/delphifaq.phtml
- www.iti.fh-flensburg.de/lang/algorithmen/sortieren
- www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci
- www.papatya.info.tr/bilisimAnsiklopedisi.htm
- www.paulgraham.com/spam.html
- www.softwarefederation.com/fractal.html
- www.techtoolblog.com/archives/195-free-online-programming-books
- www.telekom.gov.tr
- www.turkyol.com
- www.yazilimci.org/content/view/29/31

KAYNAKÇA

- BAĞRIYANIK Tarık, **Programlama Ders Notları ve Uygulamalı Genel Programlama Kitabı** (www.yunus.projesi.com)

- WALLACE Wang, **Beginning Programming for Dummies**, Wiley Basımevi, Indianapolis, 2004