T.C.
MİLLÎ EĞİTİM BAKANLIĞI

# MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

## BİLİŞİM TEKNOLOJİLERİ

## YAPISAL PROGRAMLAMA TEMELLERİ

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

# AÇIKLAMALAR

| | |
|---|---|
| **KOD** | **481BB0026** |
| **ALAN** | **Bilişim Teknolojileri** |
| **DAL/MESLEK** | **Alan Ortak** |
| **MODÜLÜN ADI** | **Yapısal Programlama Temelleri** |
| **MODÜLÜN TANIMI** | Programlama dilinde alt program yazımı, hata düzeltme ve dosyalama ile ilgili öğretim materyalidir. |
| **SÜRE** | 40/16 |
| **ÖN KOŞUL** | Akış Diyagramları modülünü almış olmak. |
| **YETERLİK** | Alt programlar ve dosyalarla çalışmaya hazırlık yapmak |
| **MODÜLÜN AMACI** | **Genel Amaç**<br>Gerekli ortam sağlandığında, alt program yazabilecek, kullanıcı dostu arabirimleri yapabilecek ve hata düzeltme yöntemleri ile hataları düzeltebileceksiniz.<br>**Amaçlar**<br>1. Alt program yazabileceksiniz.<br>2. Kullanıcı arabirimi yapabileceksiniz.<br>3. Programı test edip, hatalarını düzeltebileceksiniz.<br>4. Dosyaya bilgi yazmayı ve okumayı yapabileceksiniz. |
| **EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI** | Bilgisayar laboratuvarı ve bu ortamda bulunan bilgisayar, 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 gerçekleştiriniz. |

# GİRİŞ

**Sevgili Öğrenci,**

Her şeyden önce herkes bir programlama dilini öğrenebilir. Bilgisayarı 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 sinir bozucu olabilir ve zamanınızın da boşa geçmesine neden olabilir. Bu sebeple bu modülleri takip ederek, en az sıkıntı ve en yüksek memnuniyet ile programlamayı öğreneceğiz.

Bu modül ile kazanacağınız konular alt program yazmak, kullanıcı dostu arabirim yapmak, hata düzeltme yöntemleri ile hataları düzeltme ve dosyalamadır. Modülü bitirdiğinizde anlamadığınız yerleri tekrar okuyup uygulayınız.

Konular kapsamlı olarak, derinlemesine anlatılmamıştır. Buzdağının sadece üstünü görüyorsunuz. Ne kadar çok uygulama ve araştırma yaparsanız kendinizi o kadar geliştirirsiniz, ilerletirsiniz.

Her programlama dilinin kendine has kuralları bulunmaktadır. Bu kurallar kimi zaman birbirine benzerken kimi zaman farklılıklar göstermektedir. Kuralların anlaşılır olması her zaman programcılar tarafından istenen bir özelliktir. Siz de bu modülde çeşitli programlama dillerinden örnekler görerek işin özünü kavramanız sağlanacaktır.

Belli yerlerde geçen araştırma konuları için “Önerilen Kaynaklar” kısmından yardım almayı unutmayınız.

# ÖĞRENME FAALİYETİ–1

## AMAÇ

Programın tekrarlanan belli yerlerini alt program hâline getirebileceksiniz.

## ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Tek programlama dili ile çoğu istediğimizi yapabiliyoruz. Aynı programı diğer dillerde yazmanın amacı ne olabilir, araştırınız.
- Seri üretim ile üretilen bir malın yapım aşamalarını araştırarak arkadaşlarınıza anlatınız. Mesela, bir otomobilin üretim aşamaları gibi…

# 1. ALT PROGRAMLAR

Bilgisayarda yazılan programlar genellikle bellek ve disk gibi kaynakları en az tüketecek şekilde olan, hızlı çalışan ve az kod yazılarak yapılmış programlardır. Basit programları çoğu kişi rahat okur ve anlar. Program büyüdükçe karmaşık hâle geleceği için, en baştan planlı hareket etmek gerekir. Daha sonra zaman kaybı olmaması için önceden önlem almamızda fayda vardır.

Hikâye veya roman yazar gibi, plansız olarak program yazılmamalıdır. Basit şekilde çalışması sağlanan programa yeni eklentiler yaptıkça program karışık ve anlaşılmaz hâle gelebilir. Kodlarınız okunması zor olan çöp kodlara dönüşmemelidir.

## 1.1. Modüller Hâlinde Programlama

Programlamayı kolaylaştırmak için programcılar bazı yöntemler geliştirmiştir, bunlar:

- Sıralı komutlar,
- Dallanma komutları ve
- Döngü komutlarıdır.

"Sıralı komutları" şu ana kadar yaptığımız programlarda ifade ettik. Bu tip komutlar alt alta yazılır, en sonunda program sonlandırılır.

**Pozitif dilinde sıralı komut örneği**

```
BAŞLA
 Ad:metin;
 yazstb("İlk Pozitif Programımız");
 Ad=bilgigirişi("İsminizi giriniz ");
 yazstb("Selam, "+Ad);
SON
```

Tabii ki bir program, değişen durumlara göre başka işlemler de yapılabilmelidir. Bunu "**Eğer**" komutu ile yapabiliriz.

**Pozitif dilinde dallanma komut örneği**

```
BAŞLA
 Ad:metin;
 yazstb("Eğer kullanımı");
 Ad=bilgigirişi("İsminizi giriniz ");
 Eğer(Ad="Ali");
          {
          yazstb("Selam, "+Ad);
          }
 değilse(Ad<>"Ali");
          {
          yazstb(Ad+" sen de kimsin?");
          }
SON
```

Program içerisinde belli işlemlerin devamlı tekrar edilerek işlenmesi gerekebilir. Bunları "**Döngü, İken**" komutları ile yapabiliriz.

**Pozitif dilinde döngü komut örneği**

```
BAŞLA
i:sayı;
yazstb("Döngü komutu, çift sayılar geriye doğru sıralı");
döngü(i=10,i>=0,-2);
          {
          yazstb(i);
          }
SON
```

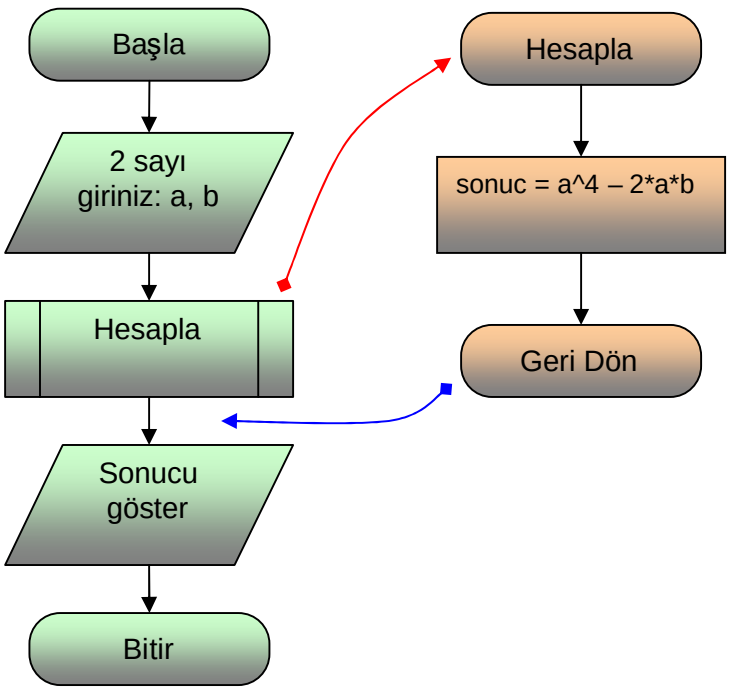
Diğer programcılar sizin programınızı anlayabilir ise, kendileri de değiştirebilirler ve geliştirebilirler. Tabi bu kural sizin için daha çok geçerlidir. Tek küme hâlinde tüm programı yazmak kumdan kale yapmak gibidir. Eninde sonunda bir yerden çökecektir.

Büyük programlar genellikle **küçük programların** (alt program da denir) birleşmesi ile yapılır. Alt programları başka programcılar yazabilir. En sonunda çalıştığından emin olunan küçük programlar kopyala yapıştır ile birleştirilir. Bunu bir evin tuğlalarına benzetebiliriz. Birine bir şey olunca tüm ev çöküp gitmez. Bozuk kısım tekrar yazılarak yerine konur. Bu yöntem ile programımıza olan güvenirliği artırırız.

Alt programlara “modül” diyebiliriz. Alt programlar hâlinde yazılarak yapılan programlamaya “**modüler programlama**” denir.

Şimdiye kadar yapılan programlar tek parça halindeki “ana program”dan oluşuyordu. Sadece bir çeşit iş, görev yapabilir hâlde idi.

Sık kullanılan komut kümelerini alt program hâline getirebiliriz. Bu alt programlar kod yazımında genellikle “ana program”ın üst kısmına yazılır. Değişken adlandırması ile aynı kurallar çerçevesinde isimlendirilir ve alt programa neler yollanacağı, alt programdan neler döneceği belirlenir. Açıklama satırları ile önemli noktaları belirtebilir, alt programların ne işe yaradığını anlatabiliriz.




**Şekil 1.1: Alt program yazmak**

1. Başla (Ana Program)
2. Oku; a, b
3. Hesapla
4. Yaz; sonuc
5. Bitir

1. Hesapla
2. sonuc = a ^ 4 – 2 * a * b
3. Geri Dön

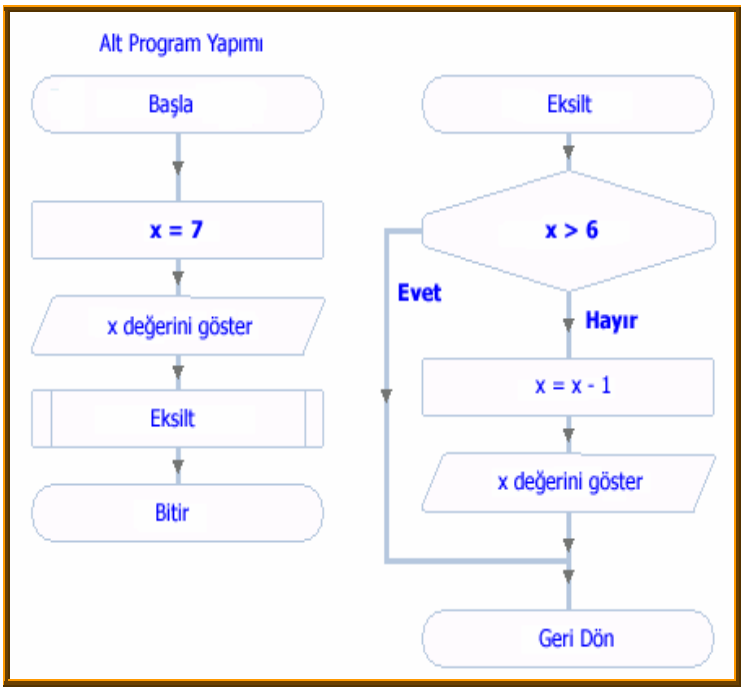
Örneklerdeki gibi alt programlar aslında ana programda olduğu gibi, belli bir başlangıç ve geri dönüş bloğu içinde olan programlardır. Alt programlar çağrılmadıkları sürece çalıştırılamazlar.

Alt programın ismini gerektiği yerlere yazıp, çağırdığımızda program akışı o program kısmına yönelir. Alt programda işlemler tamamlanınca ana program kaldığı yerden devam eder.

Alt programları ayrı dosya olarak kaydetmeyiz. Hâlâ tek dosya hâlinde, ana program ve alt programlarımızdan oluşan bir programımız vardır.

İdeal olarak bir alt program ekrana sığacak kadar uzun olmalıdır. Böylece anlaşılır, okunaklı ve hata ayıklaması kolay programlar yazabiliriz. Hatalı olan satırların tespiti bu sayede kolaylaşır.

*Not:Şekillerdeki siyah olmayan diğer mavi ve kırmızı renklerdeki okları şema çizerken sizin yapmanıza gerek yoktur. Alt programa gidiş ve dönüşü belirtmek için yapılmıştır.*




**Şekil 1.2: "Crocodile Technology" programı ile alt program yapımı**

Bazı dillerde alt program kelime olarak **fonksiyon** ile aynı anlamdadır. C dilinde sadece fonksiyon yazabilirsiniz. Basic dilinde ise **prosedür** ve fonksiyon yapımı benzer mantıkla yazılır. Terimler değişik olsa da temel aynıdır.

- **Prosedür**: Bir işlem bloğu çalıştırılır, ama değer döndürmez.
- **Fonksiyon**: İçine değer yollanır, işlenir ve kendi adına bir sonuç değeri döndürür.

## 1.2. Alt Program Tanımlama

Her alt programın yaptığı işi kısaca ifade eden uygun bir ismi olmalıdır. İçinde en az bir satır komut bulunan bu alt programlara "**kendi tanımladığımız**" komutlar diyebiliriz. Önceki derslerde gördüğümüz matematik ve metin komutları ise programlama dilinde tanımlı hazır alt programlardır.

Yazdığımız mini programlarda bazı ön değerlerin girilmesi gerekebilir. Bunu bayrak yarışındaki atletlerin birbirlerine bayrakları vermelerine benzetebiliriz. Genellikle gönderilen bilgileri parantez içine alırız.

Bir alt programa birden fazla parametre yollanabilir. Parametreler aslında bildiğimiz değişkenlerdir. İsimlendirme ve tür seçimini düzgün yapmamız gereklidir.

Klasik programlama örneği:

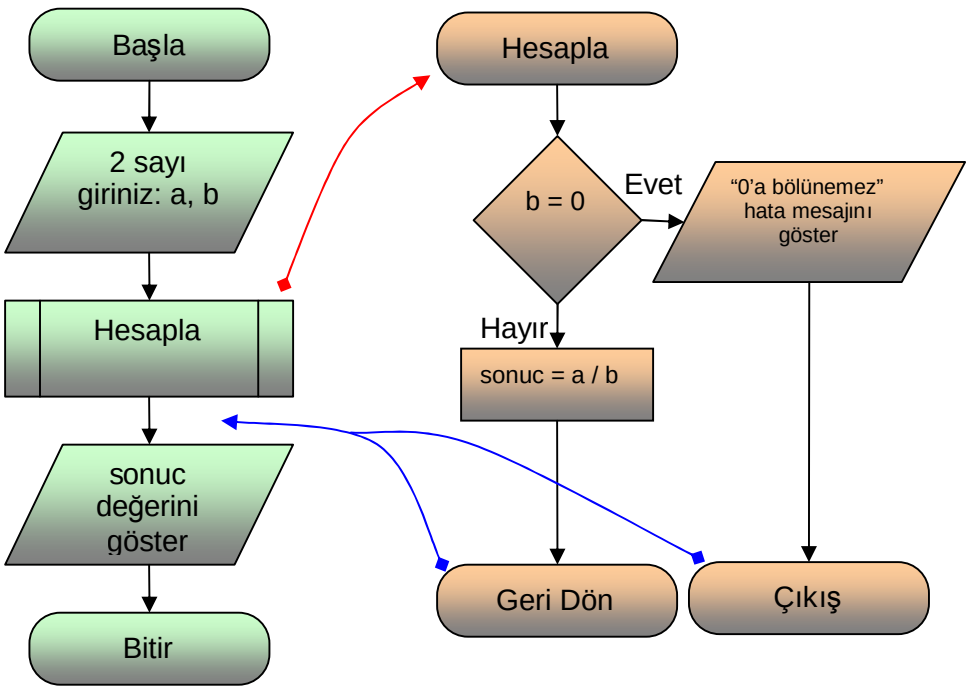
```
…
Oku; "Öğrencinin adını giriniz ", öğrAd
Oku; "Öğrencinin soyadını giriniz ", öğrSoyad
Oku; "Öğrencinin diğer bilgilerini giriniz", öğrDetay
Yaz; öğrAd & öğrSoyad & öğrDetay
…
Oku; "Velinin adını giriniz ", veliAd
Oku; "Velinin soyadını giriniz ", veliSoyad
Oku; "Velinin diğer bilgilerini giriniz", veliDetay
Yaz; veliAd & veliSoyad & veliDetay
…
```

Tekrar eden kısımları modüler (prosedür) hâle getirelim:

```
Prosedür VeriGirişi

     Oku; "Kişinin adını giriniz ", kişiAd
     Oku; "Kişinin soyadını giriniz ", kişiSoyad
     Oku; "Kişinin diğer bilgilerini giriniz", kişiDetay
     Yaz; kişiAd & kişiSoyad & kişiDetay

Prosedür Bitti


Başla
     Yaz; "Öğrenci için"
     VeriGirişi
     Yaz; "Veli için"
     VeriGirişi
Bitir
```

## 1.3. Alt Programı Kesmek ve Değer Göndermek

Döngülerde olduğu gibi, bazı şartlar meydana geldiğinde alt programın sonlanıp, ana programa dönmesi gerekebilir. Genellikle "çıkış – **exit**" komutu ile alt programlar istenilen yerden bitirilebilir.




**Şekil 1.3: Alt programdan istenmeyen ihtimalde çıkmak**

❓ Şekil 1.3'deki örnekte acaba "b" değişkeni, 0 olunca ekrana ne sonuç yazılır?

Alt program içine değer gönderme (parametreli prosedür) örneği:

```
Prosedür VeriGirişi(Metin mesaj)
     Oku; mesaj & " adını giriniz ", kişiAd
     Oku; mesaj & " soyadını giriniz ", kişiSoyad
     Oku; mesaj & " diğer bilgilerini giriniz ", kişiDetay
     Yaz; kişiAd & kişiSoyad & kişiDetay
Prosedür Bitti

Başla
     VeriGirişi("Öğrencinin")
     VeriGirişi("Velininin")
Bitir
```

Alt programdan ana programa değer gönderme (fonksiyon) örneği:

**Örnek 1:**

```
Fonksiyon VeriGirişi(Metin mesaj)
     Oku; mesaj & " adını giriniz ", kişiAd
     Oku; mesaj & " soyadını giriniz ", kişiSoyad
     Oku; mesaj & " diğer bilgilerini giriniz ", kişiDetay
     VeriGirişi = kişiAd & kişiSoyad & kişiDetay
Fonksiyon Bitti

Başla
     Yaz; VeriGirişi("Öğrencin")
     Yaz; VeriGirişi("Velinin")
Bitir
```

**Örnek 2:**

```
Fonksiyon Toplama(Sayısal sayı1, sayı2)
     Toplama = sayı1 + sayı2
Fonksiyon Bitti

Fonksiyon Bölme(Sayısal sayı1, sayı2)
     Eğer sayı2 = 0 İse
               Bölme = "Hata"
               Çıkış //alttaki bölme işlemi yapılmamalıdır
     Eğer Bitti
     Bölme = sayı1 / sayı2
Fonksiyon Bitti

Başla
     Yaz; Toplama(2, 4)   //Ekrana 6 yazar
     Yaz; Bölme(3, 0)     //Ekrana "Hata" yazar
Bitir
```

**Örnek 3:**

```
Prosedür Onaylama(Sayısal Giriş)
     Eğer Giriş <= 0 İse
               Yaz; "0 veya daha düşük rakam girilemez!"
     Eğer Bitti
Prosedür Bitti

Başla
     Sayısal sayı1, sayı2, sonuç
     Oku; "Alınan malın miktarını giriniz ", sayı1
     Onayla(sayı1)
     Oku; "Alınan malın fiyatını giriniz ", sayı2
     Onayla(sayı2)
     sonuç = sayı1 * sayı2
     Yaz; "Ödenmesi gereken fiyat ",sonuç
Bitir
```

**Örnek 4:**

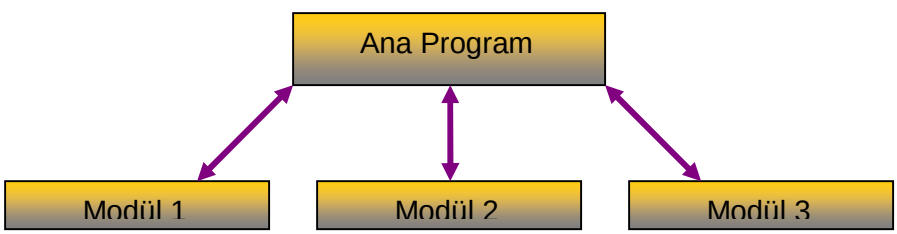
```
Fonksiyon Onaylama(Metin a, b)
     Eğer (a = "Yönetici") Ve (b = "qweasd_123") İse
               Onaylama = "Tamam" //Eğer yönetici ise onay verilir
     Eğer Bitti
Fonksiyon Bitti

Başla
     Metin isim, şifre, sonuç
     Oku; "Kullanıcı isminizi giriniz ", isim
     Oku; "Şifrenizi giriniz ", şifre
     sonuç = Onayla(isim, şifre)
     Eğer sonuç = "Tamam" İse
               Yaz; "Kayıtlara ulaşabilirsiniz"
     Değilse
               Yaz; "Kayıtlara ulaşamazsınız!"
     Eğer Bitti
Bitir
```

## 1.4. Değer Aktarımı

Alt programa veri ya "**değer**" ya da "**referans**" olarak aktarılır. Parametreler ana program ile alt program arasında veri akışını sağlayan bir köprü kurulmasını sağlar.

Değer olarak aktarmak demek, veriyi gönderen değişken ile veriyi alan değişkenin hafızada farklı yerlerde olmasıdır. Aralarında bir yalıtım vardır. Ana program ile alt program arasında tek yönde veri akışı vardır.

*Not: İki değişkenin aynı değişkenmiş gibi değerinin güncellenmesi istenir ise Basic dilinde "**ByRef, ByVal**" veya Pascal "**Var**" gibi terimleri kullanılır. C dilinde ise **işaretçiler** ile değerleri güncel tutabiliriz. Ana program ile alt program arasında çift yönlü veri alışverişi için dildeki uygun yöntemi kullanmanız gerekir.*




**Şekil 1.4: Alt programlar ve ana programın şeması**

Alt programda tanımlanan değişkenlere "**yerel değişkenler**"; kod kısmında en üstte tanımlı, tüm programda geçerli değişkenlere "**global değişkenler**" denir. Alt programlar ile global değişkenler ile de haberleşme sağlanabilir. Fakat program büyüdükçe, çok sayıda değişken tanımlamak zorunda kalınacağı için, bu daha karmaşık bir yöntem haline gelir.

Parametrelerde tanımlanan değişkenler, yerel değişkenler ile aynı ömre sahiptir. Alt program bitince, hafızadan silinirler. Global değişkenler ise program kapanınca, hafızadan silinirler. Aşağıdaki örnekte global değişkenler ile alt program ve ana programın haberleşmesi vardır.

**Örnek:**

```
      Sayısal notu
      Metin derece

Prosedür KademeHesabı
      Eğer notu > 80 İse
                derece = "A"
                Çıkış   //değer aktarımı bitince prosedürden çıkılır
      Eğer Bitti
      Eğer notu > 60 İse
                derece = "B"
                Çıkış
      Eğer Bitti
      Eğer notu > 40 İse
                derece = "C"
                Çıkış
      Eğer Bitti
      Eğer notu > 20 İse //Son ihtimalde zaten prosedürden çıkılır
                derece = "D"
      Değilse
                derece = "E"
      Eğer Bitti
Prosedür Bitti

Başla
      Oku; "Öğrenci notunu giriniz", notu
      KademeHesabı
      Yaz; derece
Bitir
```

# UYGULAMA FAALİYETİ

| İşlem Basamakları | Öneriler |
|---|---|
| 1. Sıralı olarak program komutlarını akış şemasını yapınız. | ➢ Mesela, kullanıcıdan doğum yılını isteyip ekrana yaşını bulan program yapabilirsiniz |
| 2. Dallanan programlama yöntemi ile program yazınız. | ➢ Mesela, kullanıcıya sevdiği spor türü sorulup ekrana kaç kişi ile oynandığını gösteren programı yapabilirsiniz. |
| 3. Döngü komutları ile program yazınız. | ➢ Kullanıcıdan 5 adet not girişi istenir ve ekrana bu sayıların ortalaması gösterilebilir. |
| 4. Programda sık çalıştırılan komut bloklarını bir alt programa toplayınız. | ➢ Hangi sayının çarpım tablosu ekrana çıkacağını alt program hâlinde yapabilirsiniz. |
| 5. Alt programlara uygun bir tekil isim veriniz. | ➢ İsim seçiminde programlama dilinin komutlarını (File, Not…) kullanmamalıyız. |
| 6. Bir alt programa değer yollayınız ve değer döndürünüz. | ➢ Girilen bir il plaka değerinin uygun olup olmadığı (1 ile 81 arası) değerini kontrol eden programı yapınız. |

# 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.** Bir programın bir başlangıcı vardır, birçok kere sonlandırılması olabilir. ( )

**2.** Diğer programcıların yaptığımız programlarımızı anlamasına gerek yoktur. ( )

**3.** Her programın bir ana program kısmı vardır. ( )

**4.** Hangisi ile değer döndüren bir alt program yapabiliriz?
A) Prosedür
B)Fonksiyon

**5.** İç içe komut yazımında ne yapmamalıyız?
A) Tüm komutlar aynı hizada yazılmalıdır
B) Blok başlangıç ve bitiş komutlarını yapmalıyız
C) İlk kapatılması gereken blok, en son yapılan blok olmalıdır.
D) Blok açma ve kapatma sayıları birbirine eşit olmalıdır.

**6.** Hangisi yapısal bir programlama komutu olarak kabul edilmez?
A) Döndür
B) Eğer
C) Git
D) Durum

# ÖĞRENME FAALİYETİ–2

## AMAÇ

Programınıza kullanıcı dostu (*user friendly*) arabirim yapabileceksiniz.

## ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızda kurulu programlardan hangisinin arabirimi size daha kullanışlı geliyor?
- MS-DOS pencereleri, Windows pencereleri ve web sayfalarının ara yüzlerini karşılaştırınız, farklarını belirleyiniz.
- Oyunlarda, oyuncu ile diğer nesneler (para çekme makinesi, bilgisayar, el bilgisayarı, şifreli kapılar…) arasında nasıl iletişim kuruluyor, örnek araştırınız.

# 2. KULLANICI ARABİRİMİ


Kullanıcı ile program arasında iletişim kurulmasını sağlayan ara yüze "**kullanıcı arabirimi**" denir. Genellikle programın arabiriminde bilgilendirme nesneleri ve bilgi girişi için gerekli olan nesneler bulunur. Çevremizdeki birçok araç da bir arabirime sahiptir. Mesela asansör düğme ve gösterge kısmı, merdiven otomatiği, müzik seti, televizyon, fotokopi makinesi… Bazıları çok kullanışlı ve kullanıcı dostu iken, bazıları çok karmaşık olabilir.

## 2.1. Kullanıcı Arabiriminde Pencere Tasarımı

İşletim sisteminizdeki pencereleri incelediyseniz, bazı kullanışlı noktalar dikkatinizi çekmiştir.

- Pencereler olabildiğince sade ve anlaşılır bir yazım dili ile tasarlanır.
- Her pencerenin başlık çubuğunda programın ismi ve yardımcı bilgiler yazılıdır.
- Kapat veya simge durumuna küçültmek için düğmeler vardır.
- Pencereye sığmayan nesneleri kaydırma çubukları ile denetleyebiliriz.
- Durum çubuğu ile kullanıcıya ekstra bilgilendirme sağlanır.
- Benzer görevli düğmeler sağ tarafta veya pencerenin altında yan yana hizalıdır.
- Menüler sol üst kenardan seçilebilir
- Belli yerlerde sağ tuş menüleri ile hızlı işlemler yapılabilir.
- Klavye kısayolları ile fare kullanmadan hızlı işlem yapılabilir.
- Klavye ve fare birlikte kullanılarak kolay işlem yapılabilir.
- Kritik anlarda kullanıcıdan onay istenir.
- Hata mesajları kullanıcıyı rahatsız etmeyecek şekilde sunulur.
- Fare simgeleri ile programın ne tür işlem yaptığı belirtilir.
- Kullanıcının dikkatini çekmesi için uygun renkler ve yazı tipi seçilmiştir.
- Kullanıcının uygulamayı özelleştirebilmesi için seçenekler sağlanır.

Pencerenizin (**form** da denir) boyutları ve ilk açılış yeri gözü rahatsız etmemelidir. Kullanıcı en az hareket veya çaba ile istediğini hızlı bir şekilde elde etmelidir. Fitts* kanunu bunu belirlemiştir.

Şimdiye kadar yapılan görsel programlama örneklerinde genellikle Visual Basic ve Yunus dili kullanılmıştır. Birçok programlama dili görsel arabirim yapabilmemize imkân sağlar. Sık kullanılan programlama dillerine diğer örnekler Delphi ve Visual C'dir.

İşletim sisteminden bağımsız yazılım yapmak isterseniz, Java, Python ve web programcılığı ile ilgilenebilirsiniz. Şimdilik burada Visual Basic ile arabirim oluşturma denemesi yapacağız.

## 2.2. Örnek Uygulama

Yeni bir "Windows Uygulaması" açalım. İlk adımda projenizin adı sorulur. Varsayılan isimle bırakabilirsiniz.

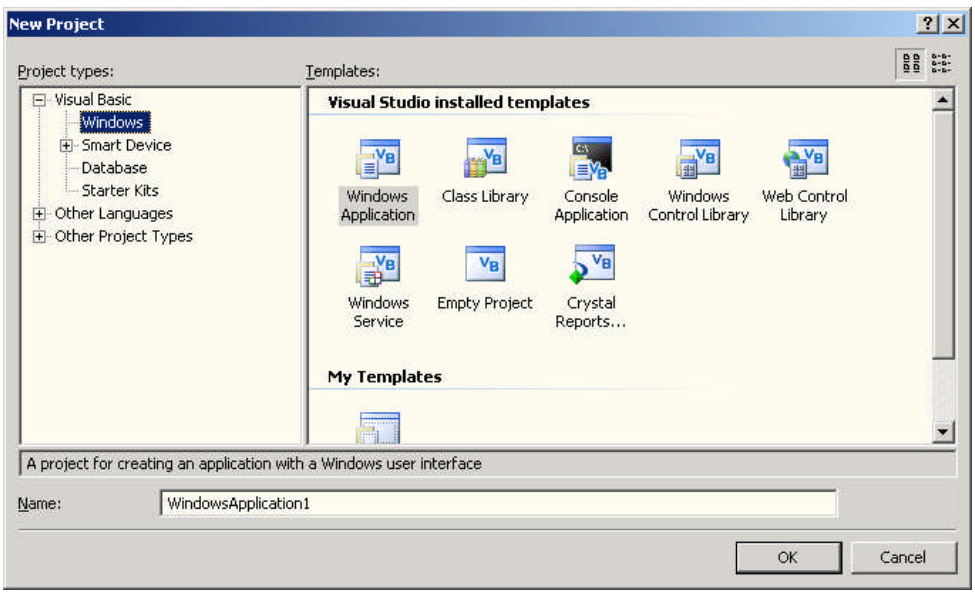
Şimdilik içinde program kodu olmayan, sadece görüntü olarak çalışan bir program yapalım. Şu ana kadar kod yazımı ile ilgili birçok örnek yaptık, görüntü ile ilgilenmemiştik.

* Paul Fitts, 1954 yılında yayımladığı ergonomi ile ilgili kitabında, bir işlemi tamamlamak için gerekli hareketleri **formül** hâlinde belirlemiştir.

İlk düşüneceğimiz konu, ne ile ilgili program yapacağımızdır. Örneğin dosya, düzen ve yardım menüsü olan basit bir not defteri tasarlayalım. Boş formu uygun bir boyuta getirdikten sonra gerekli nesneleri "Toolbox" penceresinden alıyoruz.

Nesnelerimiz:

- **Button1**: Kapat düğmesi
- **Combobox1**: Dosya türü seçmek için (içinde metin ve zengin metin yazıyor)
- **Label1**: "Dosya türü" yazısı
- **RichEdit1**: Not defterinin asıl nesnesidir, içine yazı yazılacak kısımdır
- **StatusStrip1**: "İlk programımız" yazılı
- **MenuStrip1**: Ana menüyü bununla hazırlayacağız
- **ContextMenuStrip1**: RichEdit1'e sağ tuş menüsü ekleyeceğiz

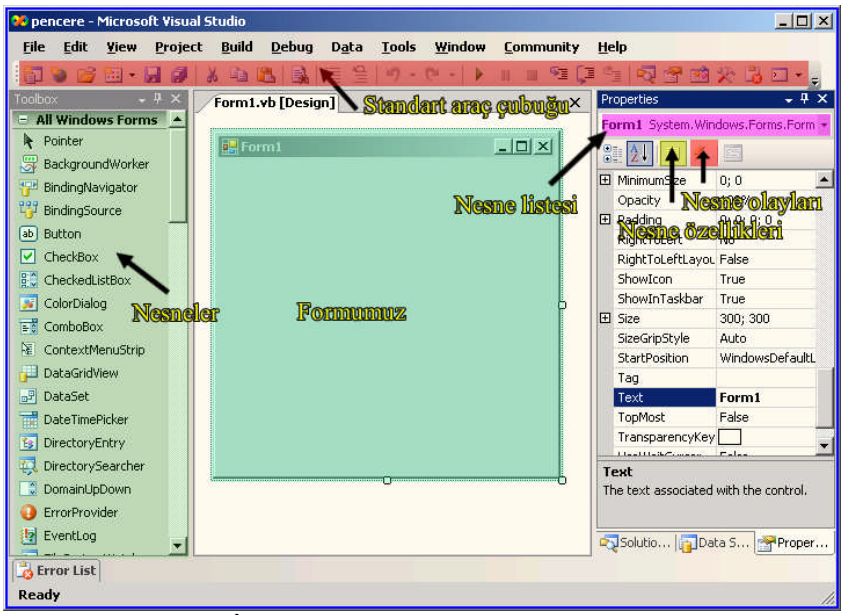



**Resim 2.1: Yeni bir proje açmak**

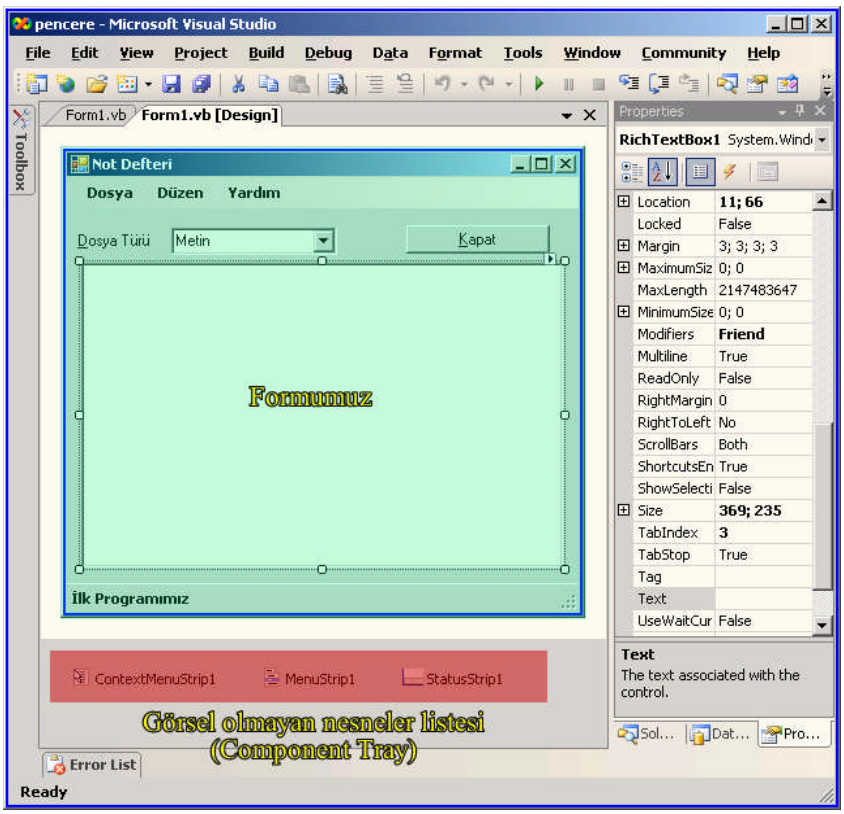
Nesneler ve özellikleri çok sayıda olduğu için burada tek tek özelliklerini anlatamayacağız. Deneme yanılma ile çok güzel özellikler keşfedebilirsiniz.

Basit bir not defterinde ne çok nesne varmış değil mi? Kim bilir kod kısmı kaç satırdır, ya da programcı kaç saat program ile uğraşmıştır. Saatlerce uğraşan programcı, programı eğer kendi isteği ile **ücretsiz** olarak yayınlıyor ise çabalarını takdir etmek, maddi ve manevi olarak destek vermek gereklidir.

Programcı, **ticari** olarak program yazıyor, ama yazılımı kaçak yollardan, kanuni olmayan yollar ile çoğaltılıyor ise, hem programcı hem de kullanıcı zarar görür. Yazılımların da diğer ürünlerdeki gibi, garantisi, destek ve yükseltme seçenekleri vardır. Kaçak olarak kullanılan yazılımda bu hizmetler yoktur. Aynı haklar müzik, kitap, resim ve film gibi üzerinde emek harcanmış ürünlerde de geçerlidir.




**Resim 2.2: İşte programınızın ilk hâli olan boş bir form**

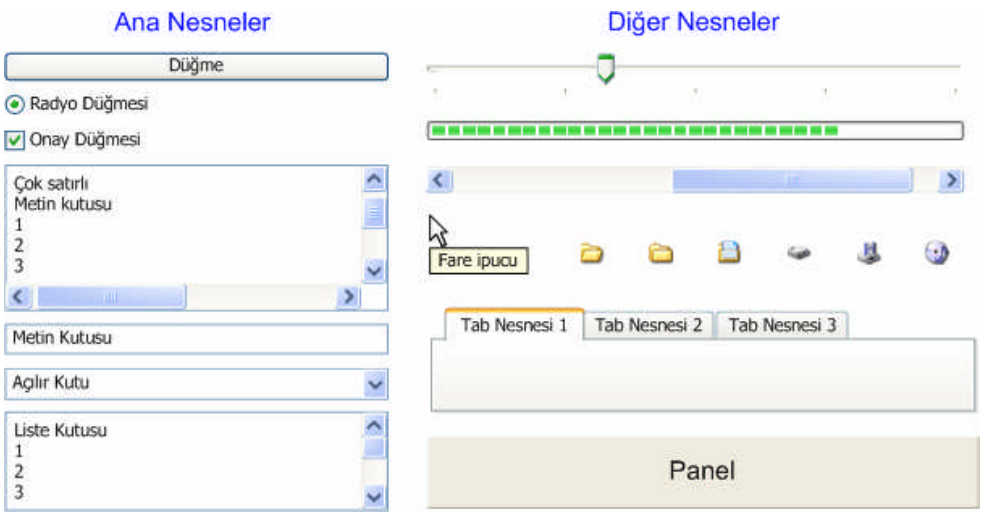



**Resim 2.3: Programın tasarım hâli**

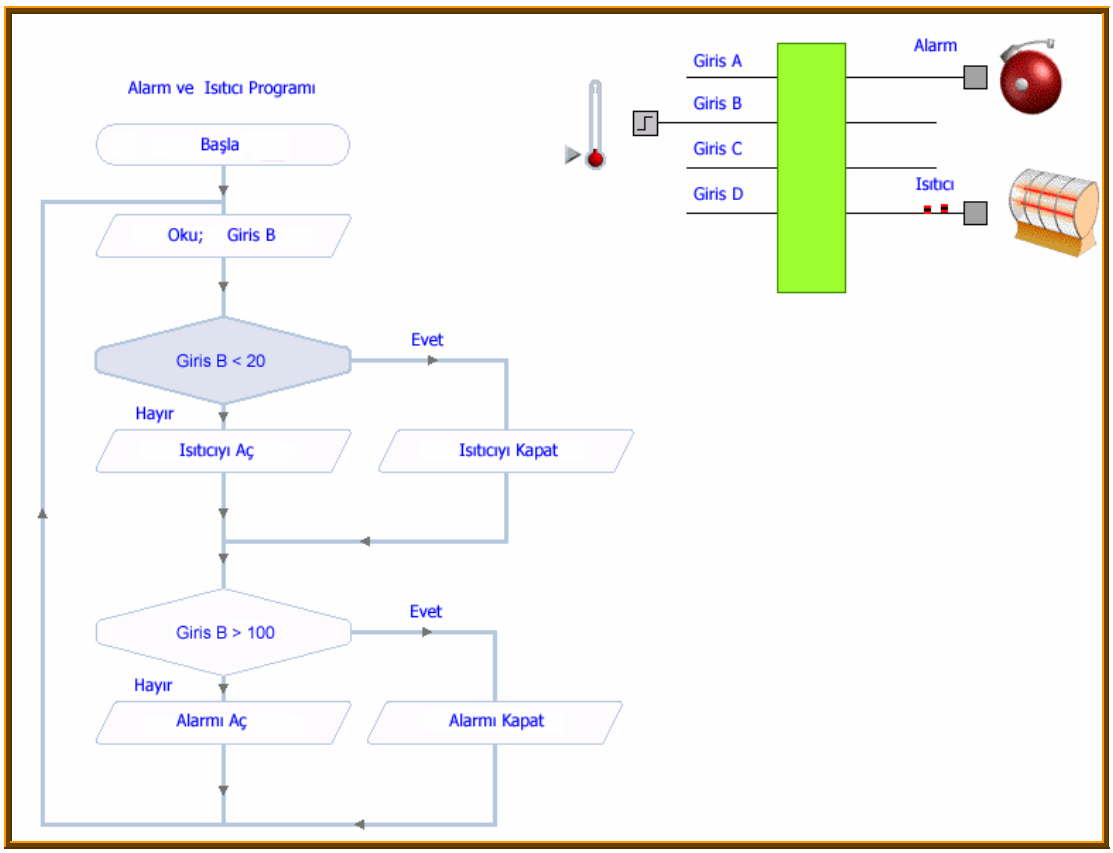

**Resim 2.4: Programımızın çalıştırıldığı an**

Görüntü olarak hazır hâle gelince artık programlama kısmına geçilebilir. Tüm nesneleri, tüm komutları kullanmanız gerekmez, zaten önemli olan bu değildir. Çok sevilen programları incelerseniz bazılarının çok sade olduğunu görürsünüz. Mesela "Bilişim Sözlüğü" gibi. Sadece aranacak kelime giriliyor ve anlamı elde ediliyor.

Metin kutusu, düğme, etiket, resim, onay kutusu, açılır kutu, liste kutusu, radyo düğmesi, menüler, grup kutusu web siteleri dâhil her programda genelde vardır.




**Resim 2.5: Sık kullanılan nesneler**




**Resim 2.6: Bir alarm ve ısıtıcı sistemi arabirimi yapımı; akış şemasını ve resmini inceleyiniz**

## UYGULAMA FAALİYETİ

| İşlem Basamakları | Öneriler |
|---|---|
| 1.Yeni bir proje açınız. | ➢ Proje ilk açıldığında boş bir form vardır. |
| 2.Pencerenin boyut ve koordinatlarını uygun hale getiriniz. | ➢ Çok büyük veya çok küçük olmamalıdır.<br>➢ Ekranın tuhaf bir yerinde açılmamalıdır.<br>➢ Diğer ekran çözünürlüklerinde nasıl göründüğünü kontrol ediniz. |
| 3.Pencere ve içindeki nesneleri uygun bir şekilde renklendiriniz. | ➢ Zıt renkler kullanınız. Siyah ile turuncu, sarı ile kahverengi gibi…<br>➢ Siyah ile gri, yeşil ile mavi çok okunaksız olabilir. |
| 4.Kullanıcıya yardımcı kısa yol menüleri tasarlayınız. | ➢ Başka bir programdaki menünün benzerini yapmayı deneyebilirsiniz. |
| 5.Pencere üzerine temel kontrolleri ekleyiniz. | ➢ Nesneleri form üzerine ekleyerek deneyiniz. |
| 6.Kontrollere program yazınız. | ➢ Bir düğmeye basit bir komut ekleyiniz. |

# 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.** MS-DOS uygulamalarında kullanıcı arabirimi yoktur. ( )

**2.** Kullanıcı arabirimi herkesin anlayacağı şekilde sade olmalıdır. ( )

**3.** Ergonomik program olmaz, istediğimiz biçimde form tasarlayabiliriz. ( )

**4.** Kullanıcı arabirimi yapımında hangisi yapılmamalıdır?
A) Kullanıcı hatalar mesajlarını görmemelidir.
B) Kullanıcıya devamlı, mesaj kutuları ile soru sormamalıdır.
C) Kullanıcı uygulamayı özelleştirebilmelidir.
D) Gizli kısayollar yaparak kullanışlılığı artırmalıyız.

**5.** Hangi dilin görsel programlama sürümü yoktur?
A) C
B) Pascal
C) Basic
D) Tupol

**6.** Hangi nesne ile yazı olarak giriş yapılamaz?
A) Metin kutusu
B) Açılır kutu
C) Yazı kutusu
D) Liste kutusu

# ÖĞRENME FAALİYETİ–3

## AMAÇ

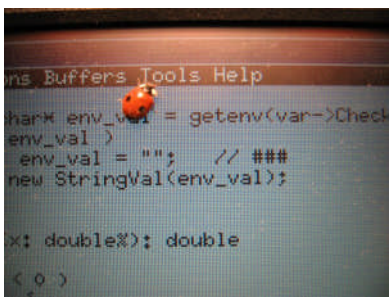
Programda oluşan hataları giderme yollarını öğreneceksiniz.

## ARAŞTIRMA

Sevgili öğrenci, bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Sadece yazılımda mı hatalar vardır? Şimdiye kadar meydana gelen donanım hatalarını araştırınız. Mesela, 1994 yılında intel Pentium işlemcilerde FDIV bölme hatası bulunmuştu.
- Bilgisayarda oluşan ilginç hata mesajlarını araştırınız. Mesela, çöken bir işletim sisteminde (mavi ekran) neler yazıyor olabilir?

# 3. HATA GİDERME




Programda bir komut yanlış yazılabilir, yanlış yere yazılabilir, bir satır yanlışlıkla iptal edilmiş olabilir. Bilgisayar bu durumda ne yapacağını bilemez ve program çöker. Problemli programlara, *böcekli* program da denir.

## 3.1. Kod Böceği Nedir?

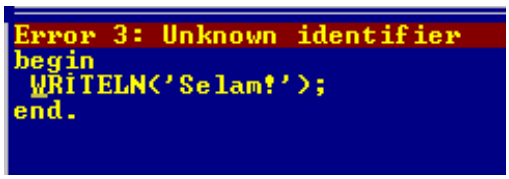
Birçok böcek zararsız görünen küçük problemlere sebep olabilirler. Asıl sorun olan böcekler, tüm programın çalışmasına engel olanlardır. Bu böcekler programın son hâlinin satış (piyasaya çıkma) tarihini erteleyebilir.

Bir böceğin oluşmasının birçok sebebi olabilir:

- **Yazım hataları**: Komutun yanlış yazılması, unutulan noktalama işaretleri gibi derleyicinin hemen bulduğu, kolay düzeltilebilen hatalardır.
- **Çalışma anı hataları**: Program hatasız derlendikten ve çalıştırıldıktan sonra meydana gelen hatalardır. Genellikle kullanıcının istenmeyen veri girmesi sonucu oluşur. Programcı önlemini önceden almalıdır.
- **Mantık hataları**: Program normalde çalışıyor gözükür, ama istenmeyen sonuç üretir. Hatayı düzeltmek veya oluşmasını engellemek, programcının programlama yeteneğine bağlıdır.

## 3.2. Yazım Hataları

Yazım hatalarını, dikkatle kodları gözden geçirerek ve program derlenirken verdiği hata mesajını iyi inceleyerek çözebilirsiniz.




**Resim 3.1: Pascal derleyicisi neden hata mesajı vermiştir?**

Derleyici genellikle hatalı yere imleci getirerek, düzeltmenizi bekler. Resim 3.1'deki gibi hatanın kodu ve türünü belirtir. Ama hata mesajı verilmeyen hatalarda ne olacak? Aşağıdaki hikâyeyi okuyunuz.

**Yazım Hatası**

1962 yılında NASA, Venüs gezegenini incelemek için "Mariner 1" adlı uydu gönderdi. Uyduyu taşıyan roket hatalı rota izlemeye başlaması üzerine, patlatılmak zorunda kalındı.

Bir hikâyeye göre, bu bir "Döngü"den kaynaklanmıştı. Normalde 3 kere dönmesi gereken döngü, basit bir yazım hatası sebebi ile hataya yol açmıştı.

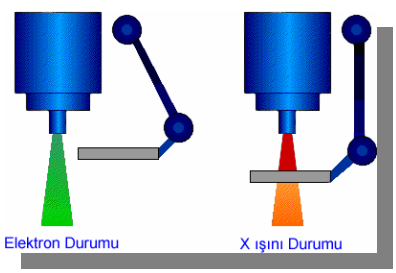
```
FOR I = 1.3  'I değişkenine 1.3 değerini atar
FOR I = 1,3  'I döngü değişkenidir, 1'den 3'e kadar döngü yapılır
```

Bu muhtemel hatalı komut kullanımı NASA'nın milyon dolarlık roketi kaybetmesine sebep olmuştu.

Sık yapılan bir hata da değişken isimlendirmesinde yapılır. Program büyüdükçe, üst satırlarda kullanılan değişken adı, yanlışlıkla diğer yerlerde farklı kullanılabilir. Garip değerlerle karşılaşırsanız değişken isimlerini kontrol ediniz. Aslında kısa ve şifreli gibi değişken adı vermemek en iyisidir.

**Sağlık**

Therac–25, radyasyon ile tedavi makinesi olarak tasarlamıştı. Hastalara yavaş yavaş radyasyon veriyordu. Aşırı radyasyon verilmemesi için yazılım ile denetlenen bir güvenlik mekanizması vardı. Ama yazılımda "ölümcül" bir hata vardı.




Makinenin 2 durumlu çalışma prensibi vardı: X ışını ve elektron ışını. X ışını durumu, yüksek seviye enerji seviyesine sahipti. Yüksek seviye enerji, elektron durumu ile dengelenmeye çalışıyordu. Yanlışlıkla X ışını durumunda kalınınca, hastalara aşırı radyasyon verilerek yanıklar ve ardından ölüme sebep olunabiliyordu.

Sonunda hata bulundu, ama birçok hasta için artık çok geçti!

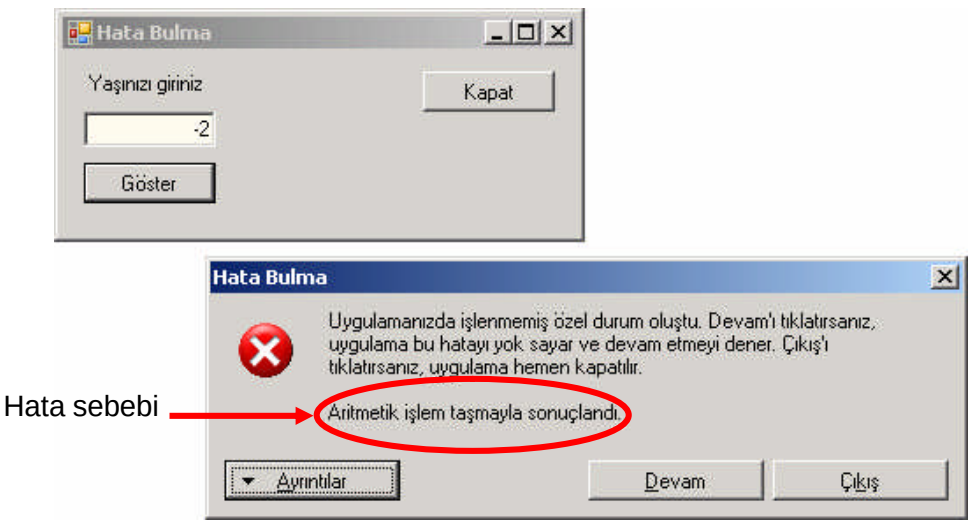
## 3.3. Çalışma Zamanı Hataları

Çalışma zamanı hataları çok sinsidir. Programcı programı iyi test etmemiş ise, hatayı kullanıcının keşfetmesine fırsat vermiş demektir. Test etmek için kendimizi kullanıcının yerine koyup, her değişiklikten sonra bol bol denemeler yapmamız gereklidir. Uç değerler denenmelidir.

Yaş bilgisinin girilmesi için 203 bin gibi bir sayı, bir de sıfır rakamını deneyin. Hatta negatif ve ondalıklı sayıları deneyin. Bakalım ne sonuçlar veriliyor:

**Visual Basic dilinde örnek**

```
Private Sub Button2_Click
  Dim yas As Byte 'Byte değişken 0-255 arası değer alır
  yas = TextBox1.Text
  MessageBox.Show("Yaşınız " & Val(yas))
End Sub
```




**Resim 3.2: Programı test değerleri ile deniyoruz**

## 3.4. Mantık Hataları

Mantık hatalarının tespiti çok zordur. Parantez hatası, unutulan karakterleri bulmak daha kolaydır. Hatta programınızı test edip çalışma zamanı hataları rahatlıkla bulunabilir.

Programı yazarken doğru kabul ettiğiniz komutlar, aslında başka şekilde çalışıyor olabilir. Bu hataları gidermek için "**Adımlama** ve **iz sürme**" teknikleri ile hatalar *yakalanmaya* çalışılır.

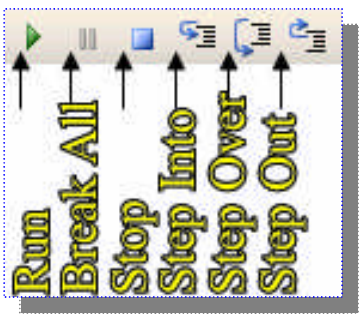
**Stratejik Hata**

1982 yılında İngiltere ile Arjantin arasında yapılan Falkland adaları savaşında, 37 yılda ilk kez İngilizler gemi kaybetmişlerdi.

Her iki taraf Fransız ürünü olan Exocet füzeleri kullanıyordu. H.M.S. Sheffield isimli gemi, üzerine gelen düşman füzelerini dost sanarak, hava savunma sistemini geçmelerine izin verdi. Gemi birçok yönden gelen füzelerle vuruldu. Kontrol odasında yangın çıktı. Mürettebattan 20 asker öldü, 27 asker yaralandı.

## 3.5. Adımlamak

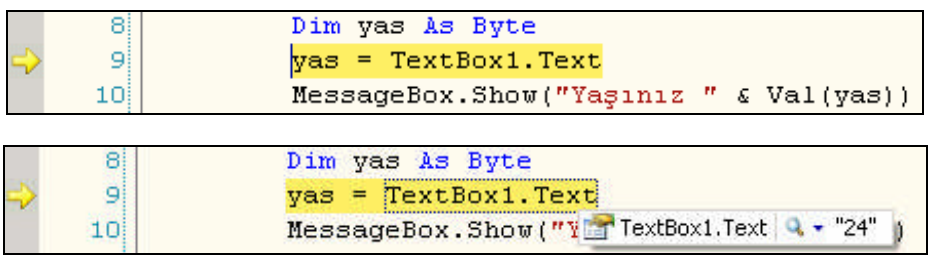
Satır olarak adımlamak programın daha iyi anlaşılmasını sağlar. Adımlama sayesinde normalde çok hızlı çalışan programı, yavaş yavaş ya da duraklatılarak izleyebiliriz.




Basic dilinde adımlama komutları:

- **Run**: Çalıştır
- **Break All**: Çalıştırmaya ara ver veya kır
- **Stop**: Programı tamamen durdur
- **Step Into**: Adım adım çalıştır
- **Step Over**: Alt programlara girmeden adımla
- **Step Out**: Alt programdan çıkıp, hızlıca ana programa dönülür, adımlamaya devam edilir

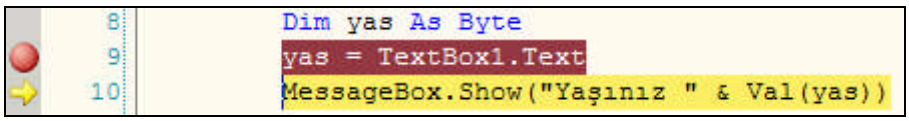
Adımlarken ve değişkenleri izlerken genellikle hatanın gözden kaçması mümkün olmaz. Adımlama sırasında nerede olduğunuzu renklendirilen satırdan anlayabilirsiniz, şu anda 9. satırda program donmuş bir şekilde bekliyor:



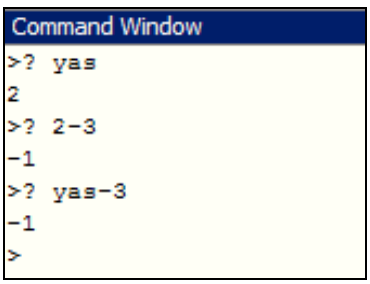

**Resim 3.3: Adımlama ve değişkenleri izleme**

Çok uzun programlarda ise hatanın olduğu tahmin edilen satıra kadar adımlamak yerine "durak noktaları" kullanılır.

- Hatanın olduğu tahmin edilen yere yakın "**durak noktası**" eklenir.[†]
- Program normal çalıştırılır.
- "Durak noktası"na gelindiğinde, programlama diline otomatik olarak dönülür ve o satır renklendirilir.
- Bundan sonrası size kalıyor, ister adımlamaya devam edin, ister "Çalıştır" komutu ile tekrar programın çalışmasına devam edin.




**Resim 3.4: Kırmızı renkli kısım "durak noktası" olan bir satırdır**



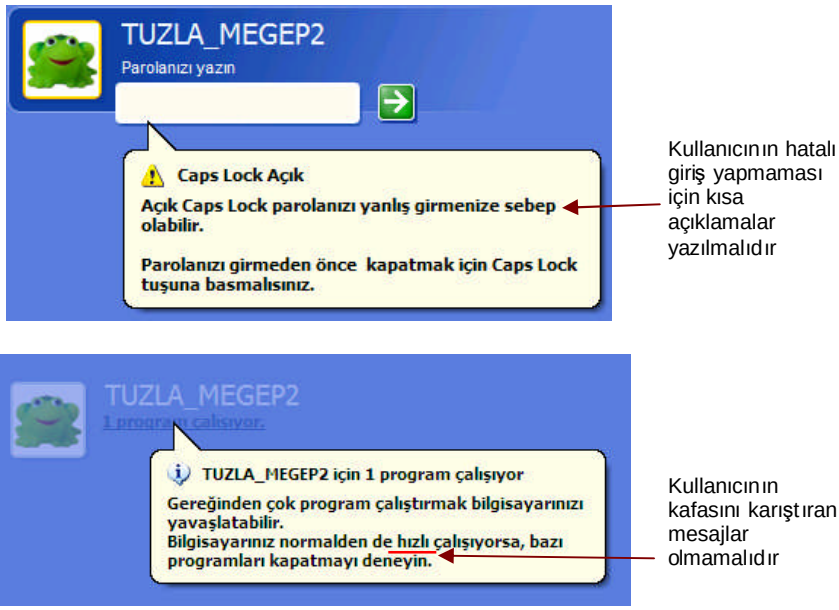

**Resim 3.5: "Command Window" kullanımı[‡]**

*Not: İsterseniz "**çalışma anında** – Run Time", değişkenler üzerinde oynama yapabilirsiniz.*

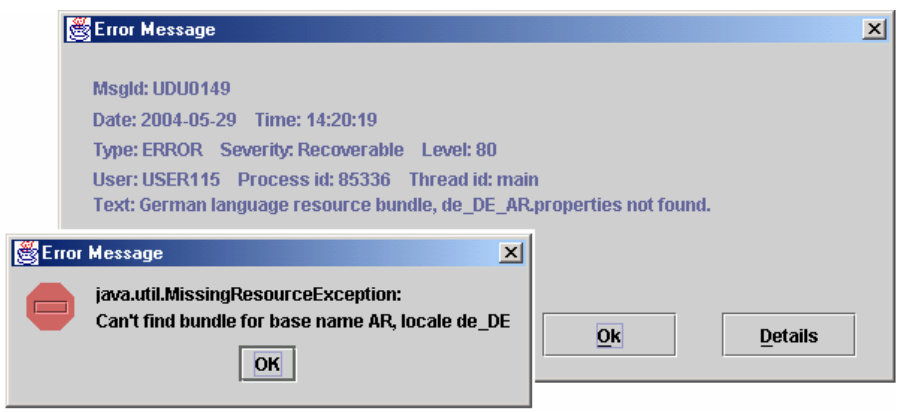
Eğer programlama dilinde böcek ayıklama imkânı bu kadar kolay değil ise, daha basit yöntemler deneyebilirsiniz. Mesela hatalı yere yakın, ekrana değişken değerini bir mesaj ile gösteren komut eklenebilir. Programdaki hata giderildikten sonra bu satırların temizlenmesi unutulmamalıdır.

Programınızdaki hataları gidermeye çalışmazsanız, yazılımınızın kalitesi düşer. Yazılımı yazan ekip kadar, test ekibinin de çok sorumluluğu vardır. Ama programdaki hatayı bulamadı diye test ekibi suçlanmamalıdır. Programcı hataları otomatik olarak yakalayan kodlar yazabilir. Kullanıcının hataları keşfetmesi firma için daha kötüdür.

[†] "Debug*Toggle Breakpoint F9" ile yapabilirsiniz
[‡] "View*Other Windows*Command Window Ctrl+Alt+A" ile pencereyi açabilirsiniz




**Resim 3.6: Kullanıcıya yardımcı olunuz, doğru bilgi vererek yönlendiriniz**

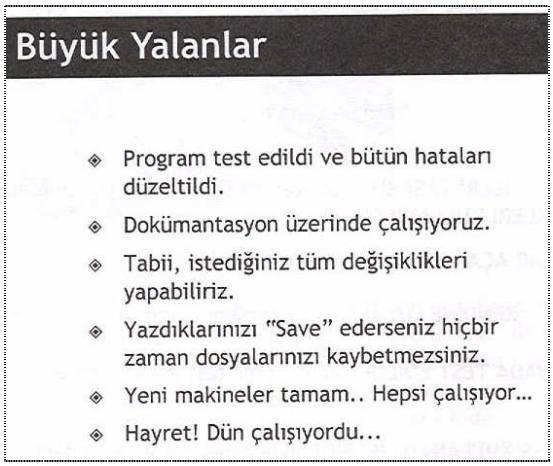



**Resim 3.7: Programcı kendisi için yaptığı basit hata ve ayrıntılı hata mesaj örnekleri**

Bir program yazarken aşağıdaki öneriler işinize yarayabilir:

- Yazılımınız ne kadar büyürse o kadar içinde **böcek** bulunma ihtimali artar.
- Böcek oluşmasını nasıl önleyebileceğinizi, otomatik olarak nasıl yakalayabileceğinizi düşünmelisiniz.
- Algoritmanızı oluştururken, kullanacağınız dili tam olarak öğrenerek koşullarınızı tamamlayınız.
- Birden fazla yol göz önüne alarak alternatifler ekleyiniz.
- Çalışma anında beliren böcek sinyallerini göz ardı etmeyiniz. Kendi kendilerine kaybolmazlar. Kaybolmasını beklerseniz daha da büyürler.
- Değişkenlerinizin değerlerini takip ediniz ve aykırı değerler almaları ne zaman ve nerede oluşuyor böcek ayıklama ile görünüz.

- Eğer programınızı kontrol amacı ile adım adım işlemekten çekinirseniz, tahmin etmediğiniz hatalarla *kullanıcı* muhatap olmak zorunda kalır.
- Denetlenmesi kolay adlandırmalar ve algoritmalar oluşturunuz.
- Hızlı çalışan program yapmak için dayanıksız kodlar oluşturmayınız.
- Kestirme ve pratik programlama risklidir. Kendi ihtiyacınıza göre çözüm üretiniz.
- Ara sıra temizlik yaparak, artık yani artan kodlarınızın böceğe sebep olmamalarını sağlayınız.
- Yaptığınız değişiklik çok küçük olabilir, ama hata oluşturabileceğini unutmayınız.
- *Sürümler* oluşturarak ilerleyiniz. Son ticari halinden hata ayıklama kodlarınızı çıkararak küçülterek ilgililere teslim ediniz.
- Program içine yerleştirdiğiniz *açıklama satırları* az ve öz olmalı ya da hiç yazmayınız.
- İmkânsız olan bir girdi olabilir mi? Kullanıcı her zaman sizin gibi düşünmez. Bu ihtimallere önlem almalısınız. Sorunları sessizce çözen kodlar yazabilirsiniz.
- Böceği gizlemekten çok onu yok etmeye çalışınız.
- Farklı algoritma yöntemleri kullanarak böceklerin sebebini çözebilirsiniz.
- Devamlı hata mesajları ile kullanıcıyı telaşlandırmayınız.
- Böceği açığa çıkarmanın yöntemi, onu bulduğunuz anda ortadan kaldırmaktır. Böcekleri tekrar çıkmaya zorlayınız.
- Çok az sayıda programcı kodlara "durak noktaları" ekleyerek, kod içinde adım adım ilerleme ve girdilerin çıktıya dönüştüğü yerlere bakma alışkanlığına sahip. Bu işlemler angarya değildir.
- Kodunuzda adım adım ilerlerken veri akışına odaklanınız.
- Arabiriminizdeki aksaklıkları bulup yok ediniz.
- Kaybolan bir böceği göz ardı etmeyin, gizleniyor olabilir.
- Zamanında bulunan böcek daha fazla böcek oluşmasına engel olur.
- Belirtiyi değil, oluşum nedenini düzeltiniz.
- Aptal böcek yoktur. Asıl sorun programcının onu bulamamış olmasıdır.
- Aynı böceğin sizi iki kere sokmasına izin vermeyiniz!



## UYGULAMA FAALİYETİ

| İşlem Basamakları | Öneriler |
|---|---|
| 1.Bir programlama dilinde program yazıp denerken, karşımıza çıkan yazım hatalarını inceleyiniz. | ➢ İlk kez karşılaştığınız hataları ve çözüm yollarını defterinizin arkasına yazınız, böylece aynı hatayla karşılaşınca bocalamazsınız. |
| 2.Programdaki yazım hatalarını düzeltiniz. | ➢ Arkadaşlarınızın yaptığı programlardaki yazım hatalarına yardım ediniz. En çok nerede hata yapıldığını, sebebini tartışınız. |
| 3.Program çalıştırıldıktan sonra denemek için veri giriniz. | ➢ Yaptığınız programları diğer insanların nasıl kullandığını gözlemleyiniz. Nasıl veri girdiklerini inceleyiniz. Sizin istemediğiniz değerleri girerlerse ekrana nasıl bir hata mesajı geliyor? |
| 4.Adım adım programı çalıştırarak, programın akışını kontrol ediniz. | ➢ Adımlama komutu ile programınızı 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.** Kod böcekleri çok akıllıdır. ( )

**2.** Derleyici tüm hataları yakalar, bize hatanın yerini gösterir. ( )

**3.** Basit bir değişiklik yapılınca, program test edilmese de olur. ( )

**4.** 1.500,45 bir sayı girişi olabilir. ( )

**5.** Hangi komut ile programı adımlayabiliriz?
A) Çalıştır - Run
B) Dur - Stop
C) Adımla - Step Into
D) Kır - Break All

**6.** Hangi hatanın tespiti ve giderilmesi çok kolaydır?
A) Yazım hataları
B) Çalışma anı hataları
C) Mantık hataları
D) Ölümcül hatalar (disk, bellek hataları gibi)

# ÖĞRENME FAALİYETİ–4

## AMAÇ

Programda saklanması gereken bilgileri dosyalara saklayabilecek ve tekrar okuyabileceksiniz.

## ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Birçok program değişik dosya türlerini çalıştırabilir ve değiştirebilir. Programların kullandığı dosya türlerini araştırınız. Mesela ofis programlarında *.doc, *.xls dosya uzantıları vardır.
- Windows işletim sisteminde çalıştırılabilen bir dosya (*.exe gibi) diğer işletim sistemlerinde de çalıştırılabilir mi? Ya da tam tersi, Linux'ta kurulabilen bir program Windows'a kurulabilir mi?

# 4. DOSYALAMA

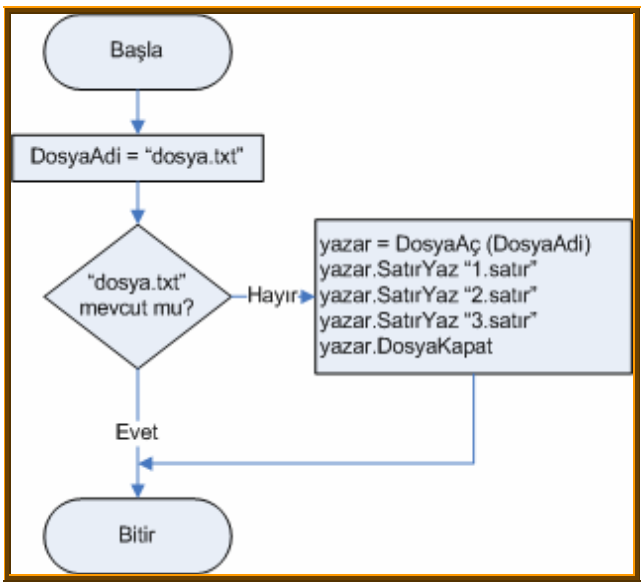
Bir önceki derste değişkenlerin ömürleri bitince hafızadan silindiğinden bahsettik. Zaten böyle olmasaydı, ana bellek birçok programı çalıştırmaya yetmezdi. Ya bize kalıcı, her program açıldığında, önceden saklanan bilginin gösterilmesi gibi bir durum gerekirse ne yapabiliriz? Oyunlardaki rekor bilgileri gibi düşünebiliriz. Bilgiler çoğunlukla ayrı bir veri dosyasında saklanır.

## 4.1. Metin Dosyalar

İsminden de anlaşılacağı gibi, **metin** dosyaların içinde sadece yazı vardır. Biçimlendirilmemiş metin dosyalarını tüm işletim sistemleri tanır, değiştirebilir.

Metin dosyalara "**sıralı dosyalar**" diyebiliriz. Bir metin dosyasını kullanabilmemiz için önce onu oluşturmamız gereklidir. Daha sonra da sıra ile satır satır metin dosyasını okuyabilir, değiştirebiliriz.

Sıralı dosya olduğu için dosyaya bilgileri sıra ile yazıp, yine sıra ile okuyoruz. Peş peşe bilgi işlemi yapıldığı için "**For** - Döndür" komutu ile daha az komut yazarak, çok satırlı bilgileri alabilir, yazabiliriz. 3 satır bilgi için basit olarak aşağıdaki gibi bir örneğimiz var.



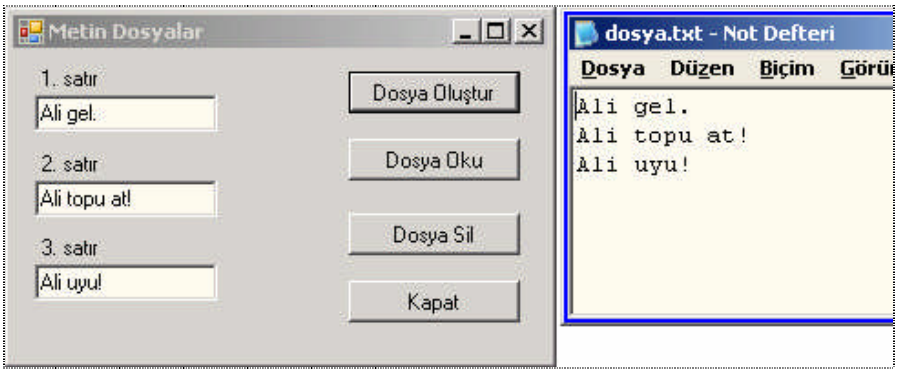

**Resim 4.1: Metin dosyaya bilgi yazma**

1. Başla
2. DosyaAdi = "dosya.txt"
3. Eğer DosyaAdi mevcut İse Git 9
4. yazar = DosyaAç (DosyaAdi)
5. yazar.SatırYaz "1. satır"
6. yazar.SatırYaz "2. satır"
7. yazar.SatırYaz "3. satır"
8. yazar.DosyaKapat
9. Bitir

Okuma veya yazma işlemi yapıldıktan sonra dosya kapatılmalıdır. Eğer kapatmayı unutursak, programda hatalar meydana gelebilir. Açık kalan dosyayı üzerinde başka işlem yapmak için tekrar açamayız. Aşağıdakine benzer bir hata mesajı oluşabilir:

```
"İşlem, başka bir işlem tarafından kullanıldığından
'C:\ornekler3\vb\metinDosyalama\dosya.txt'
dosyasına erişemiyor."
```

Metin dosyasını kendi programımız haricinde, not defterini kullanarak görüntüleyebilirsiniz. Test etmek için satırları değiştirip, programınızda tekrar okumayı deneyebilirsiniz.




**Resim 4.2: Metin dosya işlemleri ekran görüntüsü**

Programlarınızı piyasada yaygın olarak kullanılan programlar ile uyumlu yapmaya dikkat ediniz. Eğer kendi dosya türünüzü oluşturursanız, sadece sizin programınıza bağımlı hâle gelir. Mesela programınız hesap tablosu halinde verilerini kaydedebiliyor ise, hesap tablosunu kullanarak işlem yapan kullanıcıya destek sağlamış olursunuz. Programınızın kalitesi daha artar.

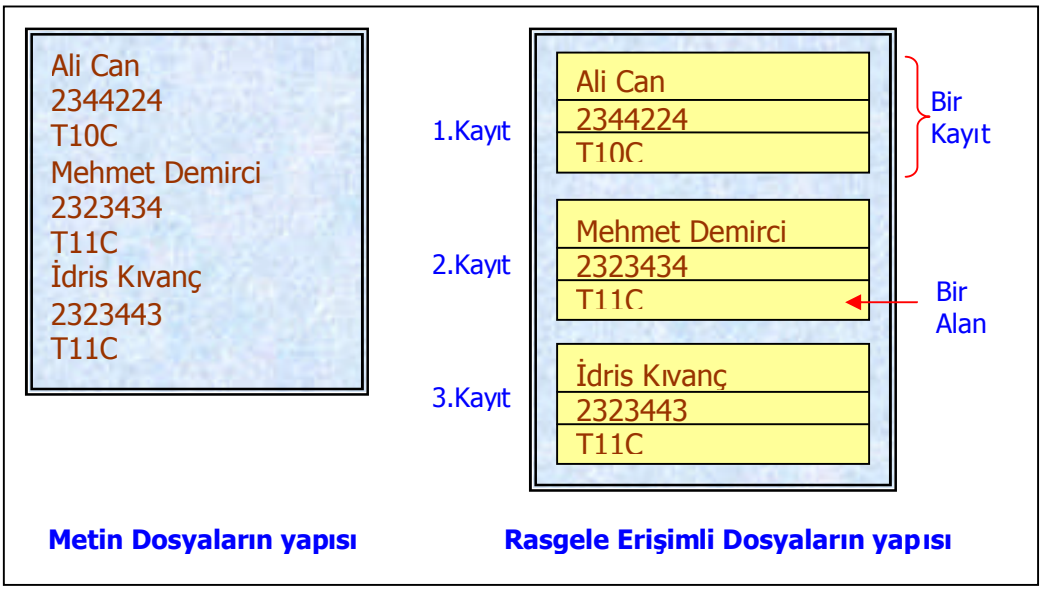
**?** Siz de programa daha fazla satır bilgi kaydetme imkânı sağlayınız. Döngüler kullanarak daha az kod nasıl yazılabilir?

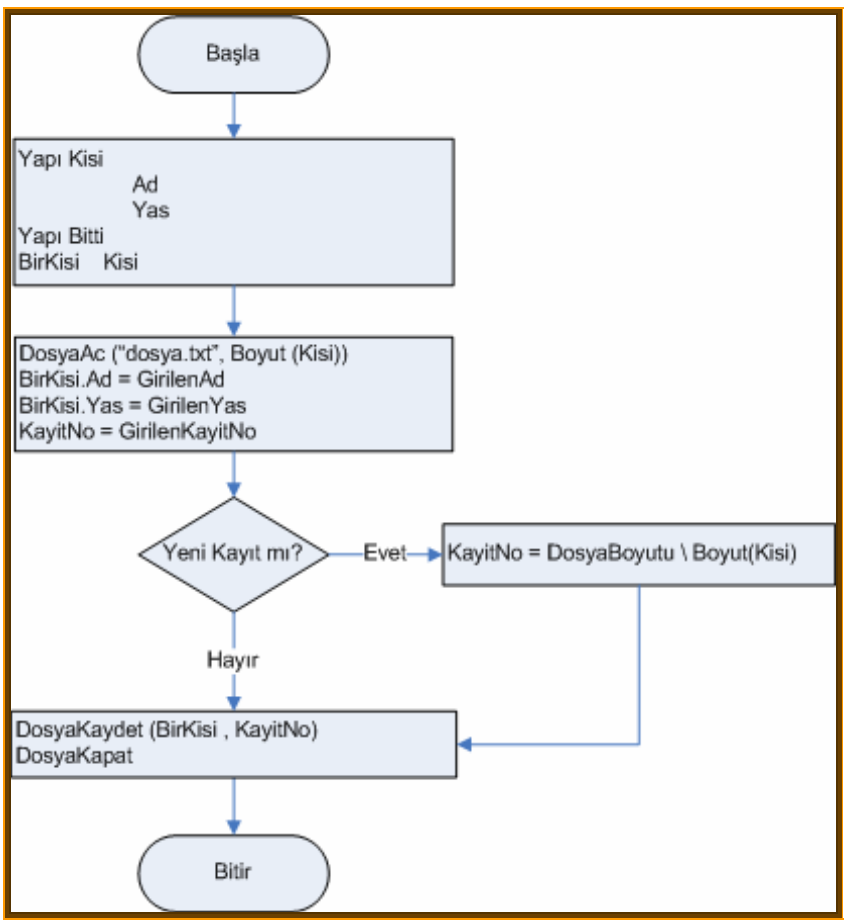
## 4.2. Rasgele Erişimli Dosyalar

Metin dosyalarda satırları tek tek okumak küçük dosyalar için hız problemi oluşturmaz iken, büyük dosyalarda belli bir satıra gidip işlem yapmak büyük problemdir. Son satırı okumak için tüm dosyayı taramanız gerekir.

Rasgele erişimli dosyalar sabit boyutlu küçük parçalar halinde "**kayıt**" isminde bölümlerden oluşur. Kayıt içinde bir veri kümesi saklanır. Her kayıt içinde "**alan**" isminde daha küçük parçalar vardır.

Kayıt uzunluklarının belirli olması sayesinde istenen bilgiye rahatlıkla gidilebilir. Bu daha etkin veri işleme anlamına gelmektedir. **Veritabanı** konusuna bu sayede girmiş oluyoruz.



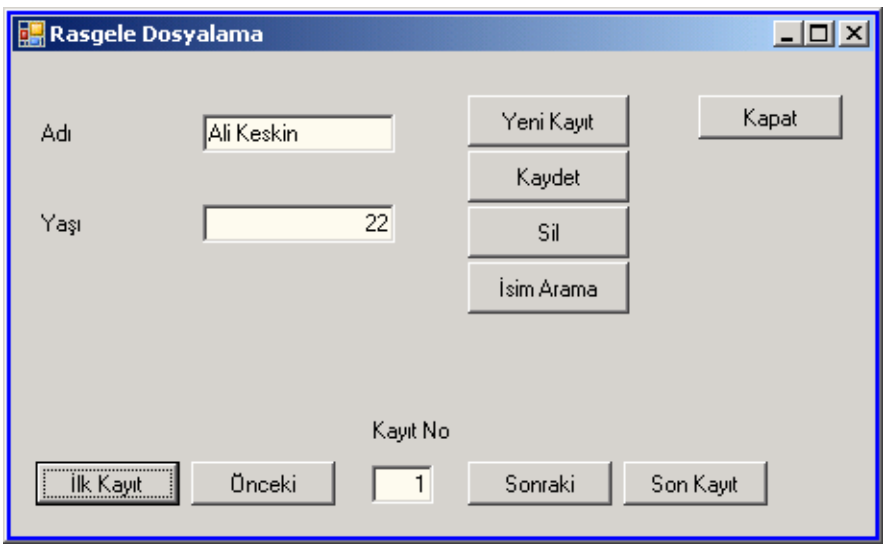




**Resim 4.3: Rasgele dosyalamada kaydetme işlemi**

Rasgele erişimli dosyada kayıt hazırlarken, ne tür verileri saklayacağınızı, kaç adet alana sahip olacağını kararlaştırmalısınız. Alanların kaç karakter büyüklüğe sahip olacağını planlamalısınız. "Daha sonradan değiştiririm" derseniz, yeni alan boyutları eskisi ile uymayacağı için veri kaybı ile karşılaşabilirsiniz.

? Silme ve isim ile arama işlemlerini tasarlayınız.

? Yaş kısmına yanlışlıkla yazı veya çok büyük bir sayı girilirse ne olur? Hatalı veri girişini önlemek için ne yapabilirsiniz?




**Resim 4.4: Rasgele dosya işlemleri**

## 4.3. İkilik Dosyalar

Rasgele dosyalamada her kaydı yapısal olarak kullanma zorluğu, programcıların yeni bir yol aramasına sebep olmuştur. Hem hızlı erişimli, hem de uzun metinleri saklayacak şekilde ikilik (*binary*) sistemde yapılan dosyalamayı kullanabiliriz. Metin uzunlukları sabit olmak zorunda olmadığından, bu yöntem dosya boyutu olarak tasarruf sağlar.

İkilik dosyalar; *.pdf dosya oluşturma, kayıt bilgileri saklama (*.log dosyaları), program ayarlarını saklama (*.bin dosyaları), internette dosya aktarımı sırasında (FTP, eposta), veritabanı içine resim, ses dosyası eklemek istendiğinde kullanılıyor.

Basit bir iş için bu kadar uzun program yazmak gereksiz görünebilir. Zaten genellikle Microsoft Access gibi "**veritabanı programları**" bizim veri saklamamıza yardım ederler. Daha karmaşık ve yoğun veri saklama ve okuma işlemleri gereken yerlerde veritabanı programları yüksek performans sağlarlar.

Bazı durumlarda ise metin dosyasına kaydetmek daha pratiktir. Mesela, sitenizin ziyaretçi sayısını saklamak, programınızın pencerelerinin en son boyutlarını saklamak için bu yöntemi kullanabilirsiniz. Çok az bilgi tutulacağı için gidip bir veritabanına bağlantı yapmaya gerek yoktur. Güvenliğin önemli olmadığı ve hız gerektiren yerlerde "metin dosyalama" ile veri saklayabilirsiniz.

**?** Dosyalara dışarıdan erişilebildiğine göre, eğer çok gizli bilgiler saklanması gerekiyor ise ne gibi önlemler almamız gerekir? Mesela evinize tek kapıdan değil de başka yerlerden girilebiliyor ise, hırsızlara karşı nasıl önlemler alırsınız?

**?** Programlar veritabanı kullanmadan başka hangi yöntemler ile ayarlarını saklar? Mesela, Windows'un kayıt defterini[§] (*registry*) kullanmak gibi …

```
Yapı Kisi
     Metin Ad
     Sayısal Yas
Yapı Bitti

Basla
     birKisi Kisi
     dosyaAdi = "dosya.txt"

     binDosya DosyaAc (dosyaAdi, AçveyaKaydet)
     yazici İkilikYazar (binDosya)

     Oku; birKisi.Ad
     Oku; birKisi.Yas

     yazici.DosyaBasinaGit
     yazici.Yaz birKisi.Ad
     yazici.Yaz birKisi.Yas

     yazici.DosyaKapat
Bitir
```

[§] "Başlat*Çalıştır…*regedit" ile kayıt defterini görüntüleyebilirsiniz.

## UYGULAMA FAALİYETİ

| İşlem Basamakları | Öneriler |
|---|---|
| 1.“DosyaAç” komutu ile dosya oluşturunuz. | ➢ Veri dosyasını metin, rasgele veya ikilik yöntemi ile oluşturunuz. |
| 2.“SatırYaz, SatırOku” komutu ile dosyaya veri giriniz veya ekrana verinin çıktısı alınız. | ➢ Bilgi yazma ve okuma için uygun komutu seçiniz. |
| 3.“Structure – Yapı” içinde alanların boyutunu, türünü belirtiniz. | ➢ Belirleyeceğiniz alanları yapı hâline getiriniz. |
| 4.“DosyaKaydet” ve “DosyaGetir” ile belli bir kayda ulaşınız. | ➢ Kaydolmuş bilgileri ekrana listeleyiniz. |
| 5.Döngü içinde verileri ekrana listeleyiniz. | ➢ Peş peşe işlemleri döngü komutları ile kolay programlanabilir hâle getiriniz. |
| 6.“DosyaKapat” komutu ile dosyayı kapatınız. | ➢ Her alt programın sonunda dosyayı kapatınız. |

# 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.** Programımız tüm dosya türlerini açıp okuyabilmelidir. ( )

**2.** Tüm dosyalar not defteri veya edit ile açılabilir. ( )

**3.** Dosya içine kod saklayıp, programımızdan kodları çalıştırabiliriz. ( )

**4.** İki program aynı anda bir metin dosyayı kullanabilir. ( )

**5.** Hangisi ile en kolay veritabanı hazırlanabilir?
A) Word
B) Excel
C) Access
D) Powepoint

**6.** Hangisini dosyalamada bir dosya uzantısı olarak seçmemeliyiz?
A) *.txt
B) *.dat
C) *.bin
D) *.doc

# MODÜL DEĞERLENDİRME

**PERFORMANS TESTİ (YETERLİK ÖLÇME)**

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 6 puan arasında olacak şeklinde değerlendirecektir.

| DEĞERLENDİRME KRİTERLERİ | Puan |
|---|---|
| Sıralı olarak program komutlarını yazma, dallanan programlama yöntemi ile program yazma, döngü komutları ile program yazma | |
| Çok tekrar eden komut bloklarını bir alt programa toplama | |
| Alt programlara uygun bir tekil isim verme | |
| Bir alt programa değer yollayınız ve değer döndürme | |
| Programa pencere ekleme | |
| Pencerenin boyut ve koordinatlarını uygun hâle getirme | |
| Pencere ve içindeki nesneleri uygun bir şekilde renklendirme | |
| Kullanıcıya yardımcı kısa yol menüleri tasarlama, pencere üzerine temel kontrolleri ekleme, kontrollere program yazma | |
| Bir programı denerken karşımıza çıkan yazım hatalarını görme, programdaki yazım hatalarını düzeltme | |
| Program çalıştırıldıktan sonra denemek için veri girme | |
| Adım adım programı çalıştırarak, programın akışını kontrol etme | |
| "DosyaAç" komutu ile dosya oluşturma | |
| "SatırYaz, SatırOku" komutu ile dosyaya veri girip, ekrana verinin çıktısı alma | |
| "Structure – Yapı" içinde alanların boyutunu belirtme | |
| "DosyaKaydet" ve "DosyaGetir" ile belli bir kayda ulaştırma | |
| Döngü içinde verileri ekrana listeleme, "DosyaKapat" komutu ile dosyayı kapatma | |
| **Toplam (en fazla 96 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 | D |
| 2 | Y |
| 3 | D |
| 4 | B |
| 5 | A |
| 6 | C |

**ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI**

| | |
|---|---|
| 1 | Y |
| 2 | D |
| 3 | Y |
| 4 | A |
| 5 | D |
| 6 | D |

**ÖĞRENME FAALİYETİ-3 CEVAP ANAHTARI**

| | |
|---|---|
| 1 | Y |
| 2 | Y |
| 3 | Y |
| 4 | Y |
| 5 | C |
| 6 | A |

**ÖĞRENME FAALİYETİ-4 CEVAP ANAHTARI**

| | |
|---|---|
| 1 | Y |
| 2 | D |
| 3 | Y |
| 4 | Y |
| 5 | B |
| 6 | D |

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

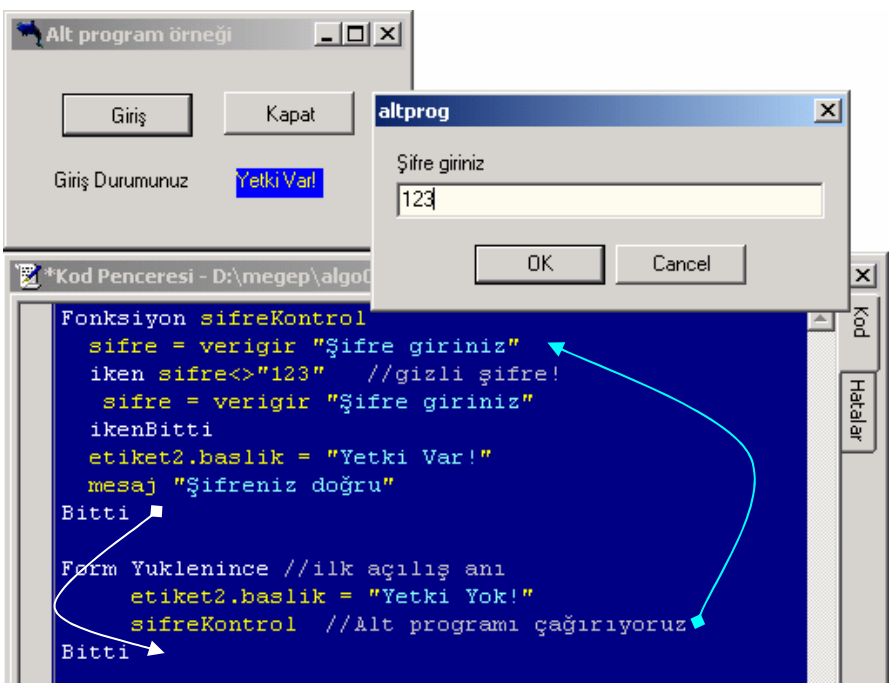
# SÖZLÜK

| İsim | Okunuş | Anlam |
|---|---|---|
| **install** | instol | kurmak, installation – kurulum |
| **instruction** | instrakşın | komut |
| **instrument** | instrumınt | enstrüman, araç |
| **integer** | intecır | tamsayı |
| **introduction** | introdakşın | tanıtım, intro |
| **item** | aytım | nesne, öğe |
| **kernel** | kernıl | çekirdek |
| **log** | log | kayıt tutmak, döküm yapmak |
| **logical** | locikıl | mantıksal, boolean |
| **match** | meç | uymak, oyun, birbirini tutmak |
| **message** | mesic | mesaj, ileti |
| **minimize** | minimayz | en küçük yapmak, simge durumuna küçült |
| **missing** | mising | eksik, kaçırmak |
| **multimedia** | maltimedya | çoklu ortam |
| **network** | netwörk | ağ |
| **object** | ıbcekt | nesne |
| **OCR** | ou si ar | Optical Character Recognition – Optik karakter tanıma, yazılımı tarayıcıların yanında verilir |
| **optimize** | optimayz | en iyi hale getirmek, optimize |
| **optional** | opşınıl | isteğe bağlı, opsiyonel |
| **page** | peyç | sayfa |
| **parameter** | perımitır | parametre, argüman, anahtar |
| **parity** | perıti | parite, eşlik |
| **partition** | partişın | disk bölümü (c: gibi) |
| **path** | pet | yol, izlek, patika |
| **pattern** | petırn | doku, dizi |
| **perform** | pörform | gerçekleştirmek |
| **peripheral** | perifırıl | çevresel birim |
| **pin** | pin | giriş çıkış için kullanılan port – veriyolu iğnesi |
| **pixel** | piksıl | Ekrandaki yazılımla elde edilebilen nokta (picture element/picture cell) |
| **pointer** | pointır | işaretçi, fare imleci |
| **port** | port | kapı, iletişim hattı |
| **present** | prezınt | var olmak |
| **previous** | priviıs | önceki |
| **primary** | praymıri | birincil |

# KOD ÖRNEKLERİ

**Tupol dilinde alt program örneği**

```
PROGRAM altProgProgrami;
TANIM
TANIMSONU
ALTPROGRAMLAR
{Alt Programimiz}
        paletCiz();
        TANIM
          Sayi i,j;
        TANIMSONU
        Basla
        Dongu j<=0 >> 199
            Blok
             Dongu i<=0 >> 319
              Blok
                   Pixel(i,j,j);
              Son;
            Son;
        Bitti;
ALTPROGRAMLARSONU

Basla
{Ana Programimiz}
 Grafikekran();
 paletCiz();
 Okugh();
  Yaziekran();
Bitti.
```




**Yunus dilinde alt program, fonksiyon (prosedür ile aynı amaçlı) yazmak**

Fonksiyon, parametre olarak değer alıp, ona göre hesap yapıp, sonuç üreten alt programdır. Genellikle "**return**" komutu ile değer geri aktarılır.

**C dilinde fonksiyon örneği**

```
#include <stdio.h>
float kupAl(float a)
          {
          return a*a*a;
          }
main()
{
float sayi = 0;
printf("Küpü alinacak sayiyi giriniz\n");
scanf("%f",sayi);
printf("Sonuc: %f",kupAl(sayi));
getch();
}
```

**PHP dilinde alt program örneği**

```
<?
//zararlı HTML kodlarını temizleme
function temizle($metin){
 $metin = str_replace("\n","",$metin);
 $metin = str_replace("\'","`",$metin);
 $metin = str_replace("\"","″",$metin);
 $metin = @strip_tags($metin);
 $metin = trim(htmlspecialchars($metin));
 return $metin;
}

$kullaniciad = temizle ($kullaniciad);
?>
```

Visual Basic'te metin dosyalarına erişim için "StreamWriter" ve "StreamReader" nesneleri kullanılabilir. Genellikle oluşturulan dosyanın uzantısını *.txt seçiyoruz.

**Pascal dilinde fonksiyon örneği**

```
uses crt;
var x,y:integer;      (* global değişkenler *)
procedure degistir(var xx:integer;var yy:integer);
var zz: integer;      (* yerel değişken *)
Begin                 (* kova yöntemi ile değiştir *)
    zz:=xx; xx:=yy; yy:=zz;
end;

begin
 clrscr;              (* ekran sil *)
 x:=10;y:=20;
 writeln(x,y);        (* ilk değerler *)
 degistir(x,y);       (* yerlerini değiştir *)
 writeln(x,y);        (* yeni değerlerimiz *)
end.
```

**Visual Basic dilinde sıralı dosya örneği**

```
Imports System
Imports System.IO

Dim DosyaAdi As String = "dosya.txt"
'*.exe ile aynı klasörde oluşacaktır

    Private Sub Button1_Click
        If File.Exists(DosyaAdi) = False Then
            Dim yazar As StreamWriter = File.CreateText(DosyaAdi)
          'ilk dosya oluşturma
            yazar.WriteLine(TextBox1.Text) 'ilk satırı yazıyoruz
            yazar.WriteLine(TextBox2.Text) 'ikinci satırı yazıyoruz
            yazar.WriteLine(TextBox3.Text) 'üçüncü satırı yazıyoruz
            yazar.Flush()
            yazar.Close()                  'dosyayı kapat
        End If
    End Sub

    Private Sub Button2_Click
        If File.Exists(DosyaAdi) = True Then
          Dim okur As StreamReader = File.OpenText(DosyaAdi)
          'okumak için dosyayı açıyoruz
          TextBox1.Text = okur.ReadLine()
          'ilk satırı dosyadan alıyoruz
          TextBox2.Text = okur.ReadLine()
          'ikinci satırı dosyadan alıyoruz
          TextBox3.Text = okur.ReadLine()
          'üçüncü satırı dosyadan alıyoruz
          okur.Close()                  'dosyayı kapat
        End If
    End Sub

    Private Sub Button3_Click
        File.Delete(DosyaAdi)
                    'tekrar oluşturabilmek için dosyayı silme
    End Sub

    Private Sub Button4_Click
        End                 'programı kapatır
   End Sub
```

**Visual Basic dilinde rasgele dosyalama örneği**

```
Structure Kisi
    <VBFixedString(15)> Public Ad As String
    Public Yas As Short
End Structure

Private Sub Button6_Click
    End
End Sub

Private Sub Button5_Click
    'kaydet düğmesi
    Dim birKisi As New Kisi
    Dim dosyaNo As Integer = FreeFile()
    Dim kayNo As Integer

    FileOpen(dosyaNo, "dosya.txt", OpenMode.Random, _
       OpenAccess.Write, OpenShare.Default, Len(birKisi))

    birKisi.Ad = TextBox1.Text
    birKisi.Yas = TextBox2.Text
    If TextBox3.Text = "*" Then
      TextBox3.Text = LOF(dosyaNo) \ Len(birKisi) + 1
    End If
    kayNo = TextBox3.Text
    FilePut(dosyaNo, birKisi, kayNo)

    FileClose(dosyaNo)
 End Sub

Private Sub Button1_Click
    'ilk kayıt düğmesi
    Dim birKisi As New Kisi
    Dim dosyaNo As Integer = FreeFile()

    FileOpen(dosyaNo, "dosya.txt", OpenMode.Random, _
       OpenAccess.Read, OpenShare.Default, Len(birKisi))

    FileGet(dosyaNo, birKisi, 1)

    TextBox1.Text = birKisi.Ad
    TextBox2.Text = birKisi.Yas
    TextBox3.Text = 1

    FileClose(dosyaNo)
End Sub

Private Sub Form1_Load
    Button1_Click(sender, e)
End Sub
```

```
Private Sub Button5_Click_1
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = "*"
End Sub

Private Sub Button4_Click
    'son kayıt düğmesi
    Dim birKisi As New Kisi
    Dim dosyaNo As Integer = FreeFile()

    FileOpen(dosyaNo, "dosya.txt", OpenMode.Random, _
       OpenAccess.Read, OpenShare.Default, Len(birKisi))

    FileGet(dosyaNo, birKisi, LOF(dosyaNo) \ Len(birKisi))

    TextBox1.Text = birKisi.Ad
    TextBox2.Text = birKisi.Yas
    TextBox3.Text = LOF(dosyaNo) \ Len(birKisi)

    FileClose(dosyaNo)
 End Sub
```

**Visual Basic dilinde ikilik dosyalama örneği**

```
Imports System
Imports System.IO

    Structure Kisi
        Public Ad As String
        Public Yas As Short
    End Structure
    Dim birKisi As New Kisi
    Dim dosyaAdi As String = "dosya.txt"

    Private Sub Button6_Click
        End
    End Sub

    Private Sub Form1_Load
        Button1_Click(sender, e)
    End Sub

    Private Sub Button8_Click
        'kaydet düğmesi
        Dim binDosya As New FileStream(dosyaAdi, _
FileMode.OpenOrCreate, FileAccess.ReadWrite)
        Dim yazici As New BinaryWriter(binDosya)
        Dim okuyucu As New BinaryReader(binDosya)
        Dim kno As Integer
        Dim Ad As String
        Dim Yas As Short

        birKisi.Ad = TextBox1.Text
        birKisi.Yas = TextBox2.Text

        If TextBox3.Text = "*" Then
            yazici.BaseStream.Seek(0, SeekOrigin.End)
        Else
            kno = 1
            While okuyucu.PeekChar() > -1
                If TextBox3.Text = kno Then Exit While
                Ad = okuyucu.ReadString
                Yas = okuyucu.ReadInt16
                kno += 1
            End While
        End If

        yazici.Write(birKisi.Ad)
        yazici.Write(birKisi.Yas)
        yazici.Close()
        okuyucu.Close()

        If TextBox3.Text = "*" Then Button4_Click(sender, e)
            'kayıt no güncelleniyor
    End Sub
```

```
    Private Sub Button1_Click
        'ilk kayıt düğmesi
        If File.Exists(dosyaAdi) Then
            Dim binDosya As New FileStream(dosyaAdi, _
FileMode.Open, FileAccess.Read)
            Dim okuyucu As New BinaryReader(binDosya)
            okuyucu.BaseStream.Seek(0, SeekOrigin.Begin)
            TextBox1.Text = okuyucu.ReadString
            TextBox2.Text = okuyucu.ReadInt16
            TextBox3.Text = 1
            okuyucu.Close()
        End If
    End Sub

    Private Sub Button4_Click
        'son kayıt düğmesi
        Dim kno As Integer
        If File.Exists(dosyaAdi) Then
            Dim binDosya As New FileStream(dosyaAdi,_
 FileMode.Open, FileAccess.Read)
            Dim okuyucu As New BinaryReader(binDosya)
            kno = 0
            Do While okuyucu.PeekChar() > -1
                TextBox1.Text = okuyucu.ReadString
                TextBox2.Text = okuyucu.ReadInt16
                kno += 1
                TextBox3.Text = kno
            Loop
            okuyucu.Close()
        End If
    End Sub

    Private Sub Button5_Click
        TextBox1.Text = "" : TextBox2.Text = ""
        TextBox3.Text = "*"
    End Sub
```

# ÖNERİLEN KAYNAKLAR


- en.wikipedia.org/wiki/Fitts'_law
- en.wikipedia.org/wiki/Pentium_FDIV_bug
- msdn2.microsoft.com
- news.bbc.co.uk/onthisday/hi/dates/stories/may/4/newsid_2504000/2504155.stm
- samples.gotdotnet.com
- www.eecs.tufts.edu/~jacob/171/slides/therac.butler.html
- www.sourgeforge.net
- www.wikipedia.org
- www.yunus.projesi.com

# KAYNAKÇA


- AYFER Can Uğur, **Kim Güler Bilgisayarlara?**, Pusula Yayınevi, İstanbul, 1998
- BAĞRIYANIK Tarık, **Programlama Ders Notları ve Uygulamalı Genel Programlama Kitabı** (www.yunus.projesi.com)
- NIIT Global Learning Solutions, Fundamentals of Programming
- WALLACE Wang, **Beginning Programming for Dummies**, Wiley Basımevi, Indianapolis, 2004