ÇUKUROVA ÜNİVERSİTESİ

MÜHENDİSLİK MİMARLIK FAKÜLTESİ

MADEN MÜHENDİSLİĞİ BÖLÜMÜ

MDZ110 Bilgisayar Programlama

(Ders Notları)

HAZIRLAYAN

Prof.Dr. Ahmet Dağ

Adana-2023

İÇİNDEKİLER **Sayfa**

1. GİRİŞ

Elle yapılamayacak kadar çok aşamalı ve işlemli karmaşık problemlerin doğru, hızlı ve az kaynak kullanımıyla çözülebilmesi için bilgisayar programlarının yazılması gerekir. Bilgisayar ancak kendisine verilen bir programı işler. Bilgisayar programları, işlemleri yapmak üzere kullanılan programlama dilinde geçerli olan deyimlerden(emirlerden) meydana gelir ve sırasıyla deyimler işletilir. Bilgisayar programlarına giriş deyimleri ile veri (bilgi) girişi yapılır ve sırasıyla işletilen deyimlerin sonucunda hedeflenilen sonuçlar yine deyimlerle hafızada saklanılır ve ekran, dosya ve yazıcı gibi birimlere aktarılır.

Karmaşık bir problemin çözümünü yapacak olan bir bilgisayar programının oluşturulması çok zaman ve emek ister fakat hazır bir bilgisayar programı ise çok sayıda kullanıcıya oldukça çok imkanlar sunar. Mevcut programlar yazarları tarafından sürekli geliştirilebilirler bu nedenle genelde programların versiyonları olur. İlaveler yapılarak geliştirilen programlar yeni versiyonları ile kullanıcıya sunulur bundan dolayı aynı programın kullanılan farklı versiyonları olabilir.

Bir problemin bilgisayarla çözümü aşağıdaki temel aşamalardan oluşur;

- Problemin tanımı,
- Problemin analizi (Algoritma geliştirme),
- Akış diyagramlarının çizimi,
- Uygun programlama dilinin seçilmesi,
- Programın kodlanması,
- Programın işletilmesi ve test edilmesi,
- Programın tanıtılması (reklamı) ve yaygınlaştırılması,
- Mevcut programın geliştirilmesi (yeni versiyonlarının hazırlanması)

1.1. Problemin Tanımı

Problemin tanımlanmasında öncelikli problemle ilgili bütün giriş ve çıkış bilgilerinin belirlenmesi gerekir. İyi tanımlanmış ve anlaşılmış bir problem yarı yarıya çözülmüş demektir. Programcı problemin temelinde yatan tekniği, bütün kuralları bilmek zorundadır.

Örneğin, bir doğrunun eğimini (yatay eksenin pozitif yönüyle yaptığı açı) belirleyen bir program yazılması düşünülüyorsa, problemle ilgili bütün kurallar şöyle yazılabilir;

- Bir doğru Y = mX+b denklemiyle tanımlanırsa, bu denklemdeki m doğrunun eğimi ve b ise doğrunun y ekseninin kestiği nokta olur.

- Eğimin açısal değeri ya bilinen doğrunun denkleminden m alınarak m = tan (α) dan α = Arctan (m) bağıntılarından hesaplanılır.

- Doğrunun bilinen iki notasının (P1, P2) koordinat değerlerine (X1, Y1, X2, Y2) göre ise de

$m = \frac{Y2 - Y1}{X2 - X1}$, α = Arctan (m) bağıntılarından hesaplanılır.

1.2. Problemin Analizi (Algoritma Geliştirme)

En önemli ve zor olan aşamadır. Öncelikle problemin çözümü için algoritma geliştirilir. Bir problemin çözülebilmesi için yapılacak işlemlerin belirli bir sıraya göre mantıksal olarak düzenlenmesi ve sözel olarak ifade edilmesi işlemine algoritma denir. Algoritmanın esas amacı kendisinden sonra gelen aşamaları yönlendirmektir. Bu aşamada problemle ilgili verilerin nereden girilip, hangi kural ve işlemlerle hesaplamaların yapılıp hedeflenen sonuçların (çıktıların) nasıl ve nereden verileceği de belirtilir.

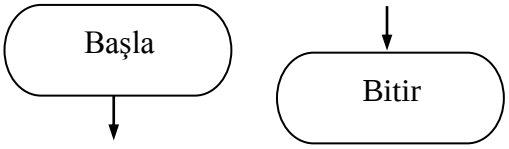
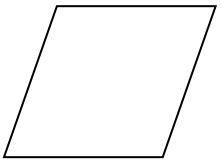
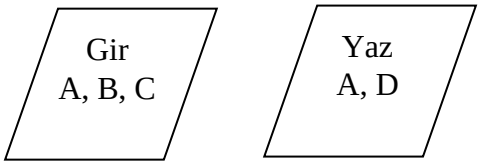
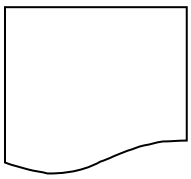
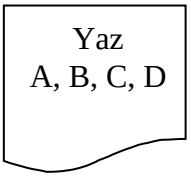
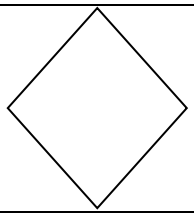
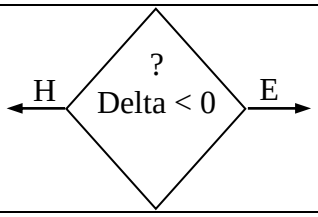
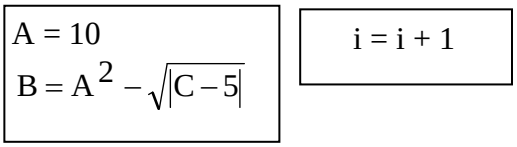
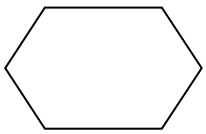
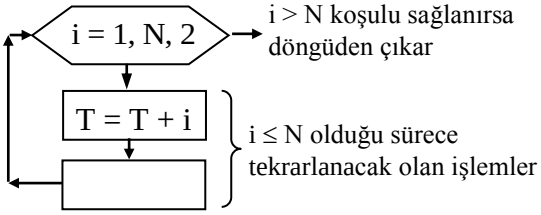
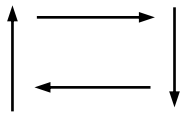
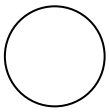
Örneğin; 100'e kadar olan pozitif çift doğal sayıların toplamını ve ortalamasını bulup yazdıran bir programın algoritması şöyle olur.

1. Aşama: Sayıyı 0 al,
2. Aşama: Çift sayıların toplamı sıfırla,
3. Aşama: Çift sayıların sayısını sıfırla,
4. Aşama: Çift sayıların toplamına sayıyı ekle,
5. Aşama: Çift sayıların sayısına 1 ekle,
6. Aşama: Sayıyı 2 arttır
7. Aşama: Sayı 100'den küçükse 4. aşamaya git, büyükse 8. aşamaya git,
8. Aşama: Çift sayıların ortalamasını bul,
9. Aşama: Çift sayıların toplamını ve ortalamasını yaz,
10. Aşama: Programı bitir.

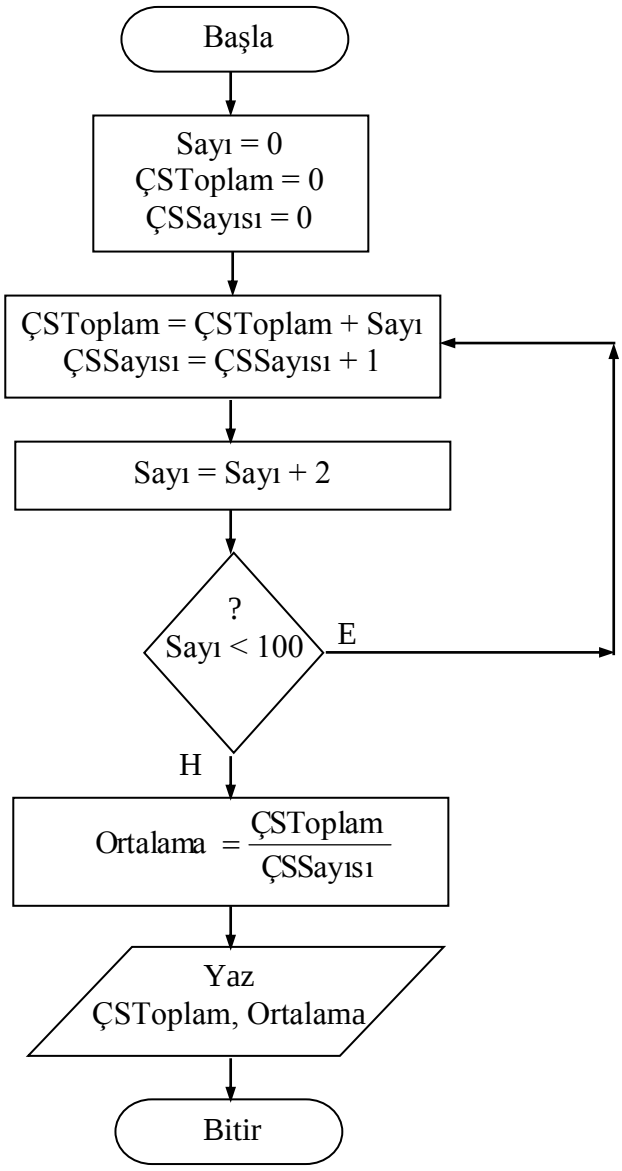
1.3. Akış Diyagramlarının Çizimi

Akış diyagramları (akış şeması, akış çizelgesi veya flowchart) algoritması geliştirilen programın aşamalarını ve işlem sırasını ifade eden özel geometrik sembollerden oluşur. Problemin akış diyagramın çizilmesi, problemin herkes tarafından anlaşılmasını ve daha kolay programlanabilmesini sağlayacaktır. Akış diyagramlarında problemi çözüme götüren bütün detaylar ve gerekli adımlar ifade edilmelidir.

Tablo 1. Akış diyagramında kullanılan semboller, anlamları ve kullanımları

Sembol	Anlamı	Kullanım örneği
	Programın başlangıcını ve bitişini belirler.	
	Genel giriş ve yazıcı dışındaki çıkış birimlerini gösterir.	
	Yazıcıdan bilgi çıkışını gösterir.	
	Koşul ve karara göre gönderme yönünü gösterir.	
	İşlem yapma veya hesaplama sembolüdür. İşlemler konuşma dilinde de yazılabilir.	
	Döngü ve çevrimi ifade eder. Belirtilen adım sayısı kadar bir aralıktaki işlemleri tekrarlar.	
	İşlem sırasını veya akış yönünü gösterir.	
	Geçiş veya bağlama işlemlerini gösterir.	

Örnek; 100'e kadar olan pozitif çift doğal sayıların toplamını ve ortalamasını bulup yazdıran bir programın daha önce geliştirilmiş olan algoritmaya göre akış diyagramı söyle olur;



1.4. Uygun Programlama Dilinin Seçilmesi ve Kodlanması (Yazılması)

Bilinen bilgisayar programlama dilleri arasında, akış diyagramı çizilerek bütün ayrıntıları belirtilmiş olan problemin yapısına uyan programlama dili seçilerek o dilin kurallarına göre genelde seçilen dilin kelime işlemci programı ile kodlar (emirler veya deyimler) satır satır yazılır. Matematiksel hesaplamalar için Fortran, grafik ağırlıklı programlar için C veya Pascal v.b. Her programlama dilinde kullanılan komutlar farklıdır.

Örnek; 100'e kadar olan pozitif çift doğal sayıların toplamını ve ortalamasını bulup yazdıran bir programın daha önce geliştirilmiş olan algoritmaya ve akış diyagramı göre bu işlemleri yapan bir hesapla() ismindeki bir altprogramın (Sub) Visual Basic programlama diliyle kodlanması söyle olur.

```
    Sub Button1_Click()
        ' ********************************************************
        ' Bu altprogram 100'e kadar olan pozitif çift sayıların
        ' toplamını ve ortalamalarını buldurur.
        '   Hazırlayan: Prof.Dr. Ahmet DAĞ
        '   Çukurova Üniversitesi Maden Mühendisliği Bölümü
        '   2004-Adana, Türkiye
        ' ********************************************************
        Sayı = 0
        ÇSToplam = 0
        ÇSSayısı = 0
10:     ÇSToplam = ÇSToplam + Sayı
        ÇSSayısı = ÇSSayısı + 1
        Sayı = Sayı + 2
        If Sayı < 100 Then GoTo 10
        Ortalama = ÇSToplam / ÇSSayısı
        TextBox1.Text = ÇSToplam
        TextBox2.Text = Ortalama
    End Sub
```

1.5. Programın Çalıştırılması ve Test Edilmesi

Kodlanarak bilgisayara girilen program kullanılan dilin işlem derleyici (compiler) tarafından derlenerek işletilir. Derleyici, seçilen dilin kurallarına ve komutlarına göre yazılan programı yazım dilinden makine diline çevirerek, mikroişlemcinin programı anlayabilmesini sağlamaktadır. Hazırlanan program sonucu önceden bilinen verilere göre denenir doğru sonuç verip vermediği test edilir. Bir programda iki türlü hata ile karşılaşılır;

- Yazım (SYNTAX) hataları: Dillerden güncel olanlar kullanılıyorsa bu hatalar editörde yazılırken varsa yazım hataları için uyarılar verir. Bu uyarılara göre hata kolaylıkla giderilir.
- Mantık hataları: Hatalı işlem sırası veya kuralların yanlış girilmesi mantık hatalarına neden olur ve yanlış sonuçları doğurur. Hiçbir derleyici mantık hatalarını yakalayamaz.

1.6. Programın Tanıtılması (Reklamı) ve Yaygınlaştırılması

Herhangi bir bilgisayara yüklenip çalıştırılabilecek şekle getirilen programın kurulumu ve kullanımı hakkında gerekli açıklamaların yer aldığı tanıtım ve kullanım kitapçıkları hazırlanabileceği gibi, gerekli açıklamalara program içerisinde den de erişilebilecek şekilde yardım menü veya dosyalarıyla da verilebilir. Geliştirilen program kullanıcılarına bir şekilde tanıtımı yapılarak kullanılmak üzere satışa sunulur.

1.7. Mevcut Programın Geliştirilmesi (yeni versiyonlarının hazırlanması)

Çok yaygın ve farklı alanlarda kullanılmak üzere yazılan kapsamlı bir program hiçbir zam en iyi haliyle oluşturulamaz. İlk kullanımı için Beta versiyonu oluşturulur hitap ettiği kullanıcılardan gelen öneriler doğrultusunda ve kullanılan dilin yeni özellikleri de kullanılarak programın yeni versiyonları geliştirilir ve kullanıma sunulur.

2. VİSUAL BASİC PROGRAMLAMA DİLİ

Görsel olmayan(Dos tabanlı) Basic dili ile program yazılırken GWBASIC ve QBASIC editörleri kullanılırdı. Daha sonra Visual Basic görsel programlama dili geliştirildi. Visual Basic dilinin sürümleri sırayla;

Visual Basic 1.0,

Visual Basic 3.0,

Visual Basic 4.0,

Visual Basic 6.0,

Visual Basic .NET

-VB 7.0 (VB 2002)

-VB 7.1 (VB 2003)

-VB 8 (VB 2005)

-VB 2008

-VB 2010

-VB 2012

-VB 2013

Şu anda Visual Studio 2013 sürümü kullanımdadır. Bu ders kapsamında anlatılacak konular kapsamında VB 2008 ile son sürümü arasında herhangi bir fark bulunmamaktadır. Burda anlatılacaklar VB 2008 sürümü üzerinde yapılacaktır.

Visual Basic (VB) 2008 Express Edition(VBEE)

Visual Basic 2008 Express Edition bir programlama arabirimidir. Kullanıcılar için arayüz geliştirme, kod yazma, yazılan kodun hatalarını ayıklama, derleme vb. fonksiyonlarıyla yazılımcıların çözüm geliştirmelerini sağlar ve Visual Studio 2008 üzerinde çalışır. Profesyonel programcıların program geliştirme yöntemi olarak kullandıkları Nesne Yönelimli Programlamayı tam olarak destekleyen Visual Basic 2008, profesyonel uygulamalar geliştirmek için kullanılan bir dildir. VBEE de bu dilin tüm özelliklerini destekler. Diğer taraftan sihirbazlar, sürükle-bırak, hazır programcıklar ve akıllı etiketler sayesinde çok kısa zamanda, çok daha az kod yazarak başarımı yüksek uygulamalar geliştirmeye zemin sağlar.

VB 2008 Express Edition, Microsoft firmasının,

http://www.microsoft.com/express/vb/default.aspx internet adresinden ücretsiz olarak indirilebilmektedir.

İndirilen dosyanın içinde;

- VB 2008 Express Edition,
- .Net Framework 3.5,

 MSDN Express Library 2008,

 Microsoft SQL Server 2008 Express Edition bulunmaktadır.

Bu programların istenirse hepsi aynı anda bilgisayarınıza kurulabileceği gibi, sadece bir veya birkaçı da, o anda veya daha sonra kurulabilir.

Bu sürümü çalıştırabilmek için bilgisayarınızda bulunması gereken asgari şartlar şunlardır:

İşlemci: 600 MHz veya daha hızlı (Tavsiye edilen: 1 GHz veya daha hızlı).

RAM: Minimum: 192 MB (Tavsiye edilen: 256 MB, eğer VBEE ile birlikte SQL Express yüklendiyse, 512 MB veya daha fazlası).

Hard Disk: Sabit diskinizde en az 500 MB boş yeriniz olmalı.

Tam kurulum: VBEE kurulum programının tamamını (VB 2008 Express Edition, .Net Framework 3.5, MSDN Express Library 2008, Microsoft SQL Server 2008 Express Edition) bilgisayarınıza yüklemek isterseniz, sabit diskinizde 1.3 GB boş alana ihtiyacınız olacaktır.

Uyarılar: VBEE'nin bilgisayarınızda çalışabilmesi için, .NET Framework 3.5 bilgisayarınıza kurulu olmalıdır.

Bir projede bulunabilecek dosyalar ve uzantılarının anlamları:

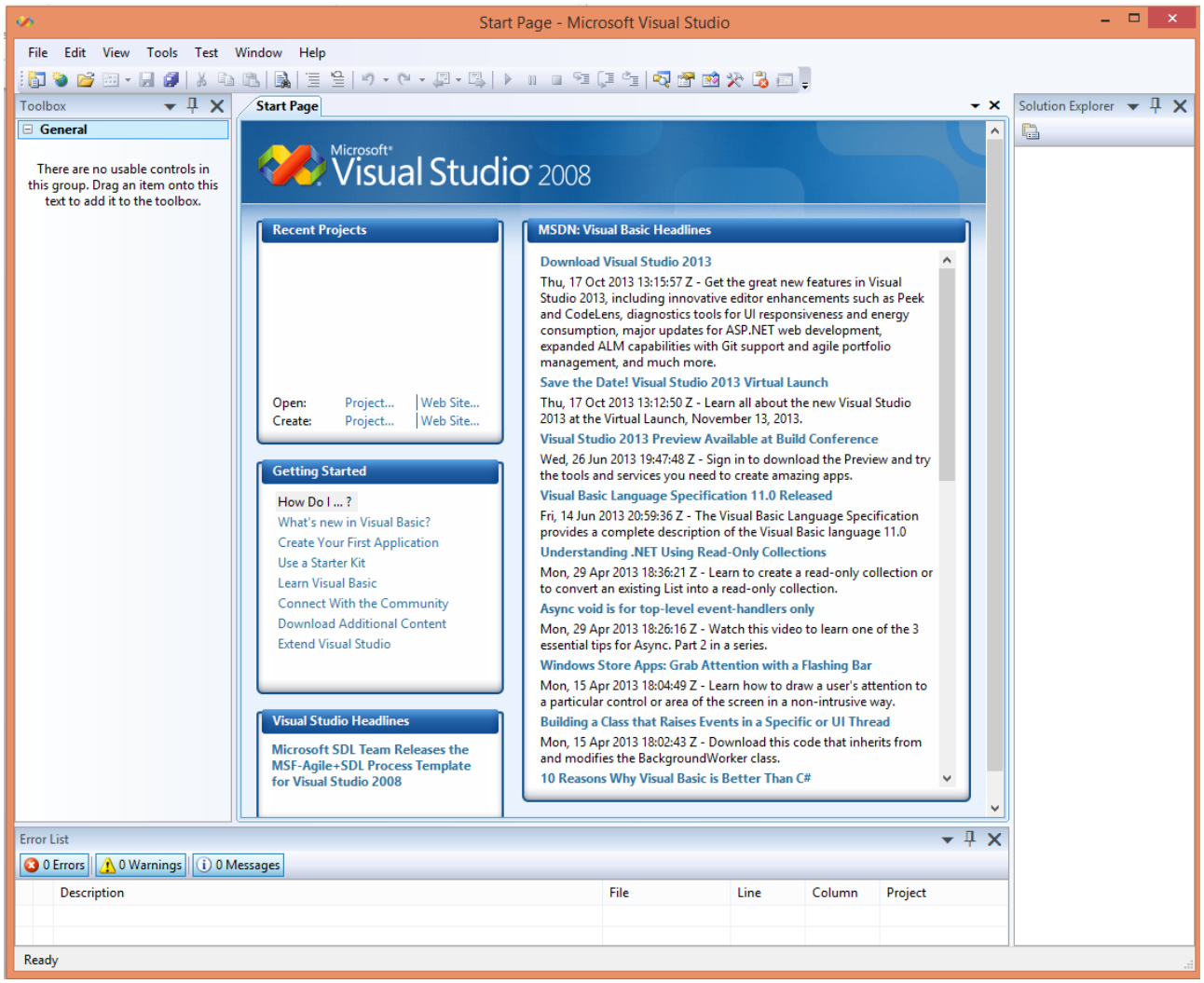
Dosya İsmi.uzantısı	Dosyanın anlamı
Solution İsmi.sln	Solution dosyası
Solution İsmi.suo	Solution seçimlik dosyası
Proje İsmi.vbproj	Proje dosyası – Çözümdeki her bir proje için bir tane
Proje İsmi.vbproj.user	Bir başka proje dosyası – Çözümdeki her bir proje için bir tane
Form İsmi.Designer.vb	Form designer dosyası – her bir form için bir tane
Form İsmi.vb	Formu oluşturan program satırları-kod satırları dosyası – her bir form için bir tane
Form İsmi.resx	Form kaynakları dosyası – her bir form için bir tane

Programın çalıştırılması:

Eğer VBEE kurulumunuzu problemsiz bir şekilde gerçekleştirdiyseniz;

Başlat-Programlar-Microsoft Visual Basic 2008 Express Edition

Visual Basic 2008 program çalıştırıldığında sonradan açılış ile ilgili bazı değişiklikler yapmadıysanız (bu işlemi; VBEE editörüne girdiğinizde, Tools-Options (bu pencerede en alttaki "Show all settings" seçili olmalı)-Environment-Startup-At startup yolu ile değiştirebilirsiniz) büyük bir ihtimalle aşağıdaki açılış ekranı ile karşılaşırsınız.



Bu ekrandaki;

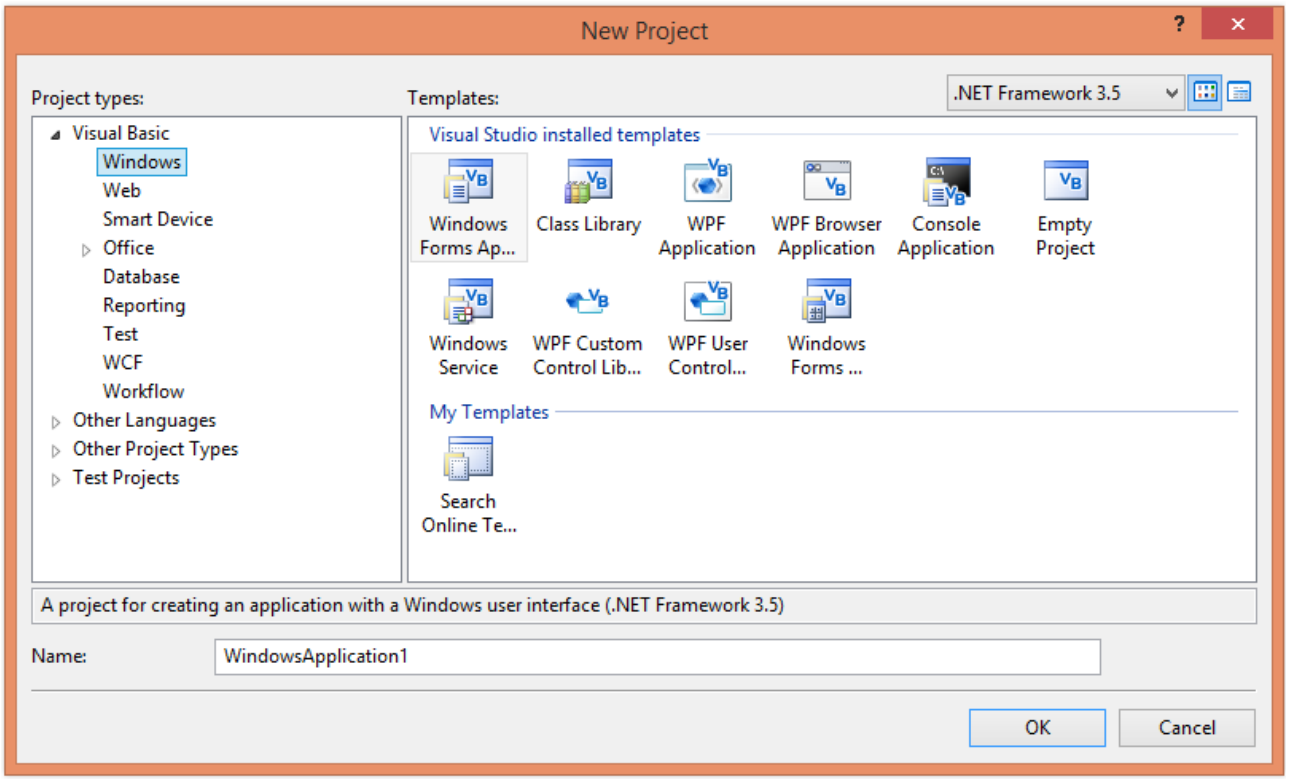
"**Recent Projects**" pencerindeki "**Open**" ifadesinin yanındaki "**Project**" seçeneği; daha önce üzerinde çalıştığınız bir VBEE projesini tekrar açmak için kullanılır.

"**Create**" ifadesinin yanındaki "**Project**" seçeneği; ilk defa oluşturacağınız bir VBEE projesini açmak için kullanılır.

Diğer pencerelerde, VBEE ile ilgili son haberler, yeni kullanıcılar için yardım bölümleri vbg. Linkler bulunmaktadır.

Yeni bir projeyi başlatma:

Açılış ekranından "Create Project" seçeneği tercih edilirse, aşağıdaki açılış ekranı ile karşılaşılır.



Windows Applications: Bu derste Windows Uygulamaları üzerinde çalışılacağından, Tempalates kısmından "Windows Applications" seçeneği tercih edilmeli ve pencerenin altındaki "Name" kutusunda (istenirse) projeye bir isim verilmelidir (projeye isim verme işlemini bu pencerede yapmak zorunda değilsiniz, isterseniz sonraki adımlarda da bu işlemi gerçekleştirebilirsiniz).

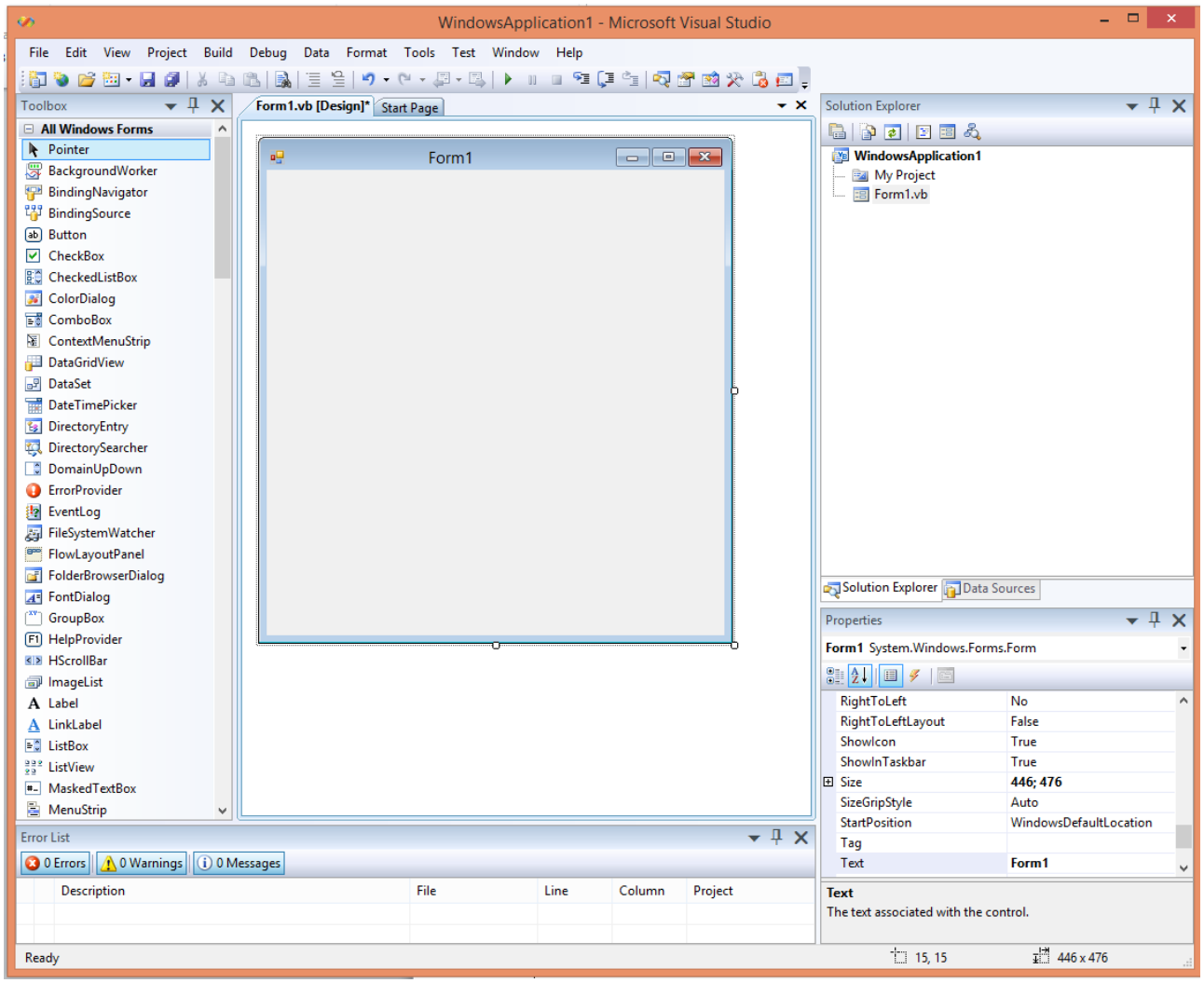
VBEE Çalışma Ortamı:

Çalışma ortamında üç halde bulunabilirsiniz;

1-Design modu: Projenizi geliştirme, tasarlama durumunda çalıştığınız ortamdır (aşağıdaki ekran görüntüsündeki; Form1.vb [Design] sekmesine dikkat edin, bu ifade kullanıcının şu anda tasarım (design) durumunda olduğunu göstermektedir).

2-Run modu: tasarladığınız projeyi çalıştırdığınız durumdur.

3-Debug modu: projenizi çalıştırma sırasında, sonlandırmadan (proje sonlanmadan) hataları ayıklamak (bulmak) için bulunulan durumdur (projenizin sonlanmadan hangi satırda duracağını siz belirlersiniz, bu durum projenizden uygun değerleri elde edemediğiniz durumlarda ve özellikle "programın hangi aşamasında hangi değerler elde ediliyor" sorusu için çok kullanışlıdır).



Form Penceresi: Projemizi tasarladığımız penceredir (nesne-kontrol-obje'dir), dolayısıyla projeyi oluşturan nesneleri üzerinde bulundurur. Aksi belirtilmediği sürece projemizi çalıştırdığımızda karşımıza gelecek ilk ekrandır. Bundan dolayı bu pencerenin üzerine yerleştirilenler aynı zamanda başlangıç görüntümüz olur. Form penceresine, ekranın sağındaki "Solution Explorer" penceresindeki Form1.vb ifadesine tıklanarak da ulaşılabilir.

Toolbox Penceresi: Projemizi tasarlarken kullanabileceğimiz nesneleri üzerinde bulunduran penceredir. Toolbox üzerinde bulunan nesneler form penceresine taşınarak projeler oluşturulur. Yeri gelmişken şunu da belirtelim; Nesne-Obje- Kontrol ifadeleri VBEE'de benzer ifadelerdir ve VBEE nesne yönelimli (object-oriented language) bir programlama dilidir.

Properties Penceresi: Form penceresine taşınan nesnelerin özelliklerini ve olaylarını (event) üzerinde bulunduran penceredir. Bu pencere sayesinde projemizde kullanacağımız nesnelerin; boyutları, rengi, üzerinde bulunmasını istediğimiz yazı, büyüklüğü, resmi vbg. özelliklerini

değiştirebiliriz ve sözkonusu nesnenin hangi olaylarının kullanılabilir olduğu ve hangilerinin kullanıldığını Properties penceresi sayesinde görebiliriz.

Solution Explorer: Projeyi oluşturan bütün program parçalarını üzerinde bulunduran ve bu programlara istediğimiz anda ulaşmamızı sağlayan penceredir. Projede kaç tane form, module, sınıf vs. kullanılmış, bu pencere sayesinde öğrenebiliriz.

Önemli tanımlamalar veya temel bilgiler;

Olay (Event): VB Olay Sürümlü (Event-Driven) bir programlama dilidir. VB'de Olay için tıklama, yazı kutusuna bir harf yazma örnekleri verilebilir. Olay Sürümlü ifadesi, olay işlemi olmadan Windows'un hiçbirşey yapmadan kullanıcıyı beklemesi anlamındadır.

Proje: VBEE programının kodları (program satırları), formları, kontrolleri, nesneleri, yardımcı program parçaları vbg. olan bütün bileşenleridir. Yani bir VB programından bahsederken aslında bir VB projesinden bahsetmiş oluyoruz.

Denetim (Control): Nesneler yaratmak için kullanılan araçtır. Araç kutusunda bulunurlar.

Nesne (Object): Denetimler ile yaratılan kullanıcı arabirim öğelerinin adıdır. Formun kendisi de bir nesnedir.

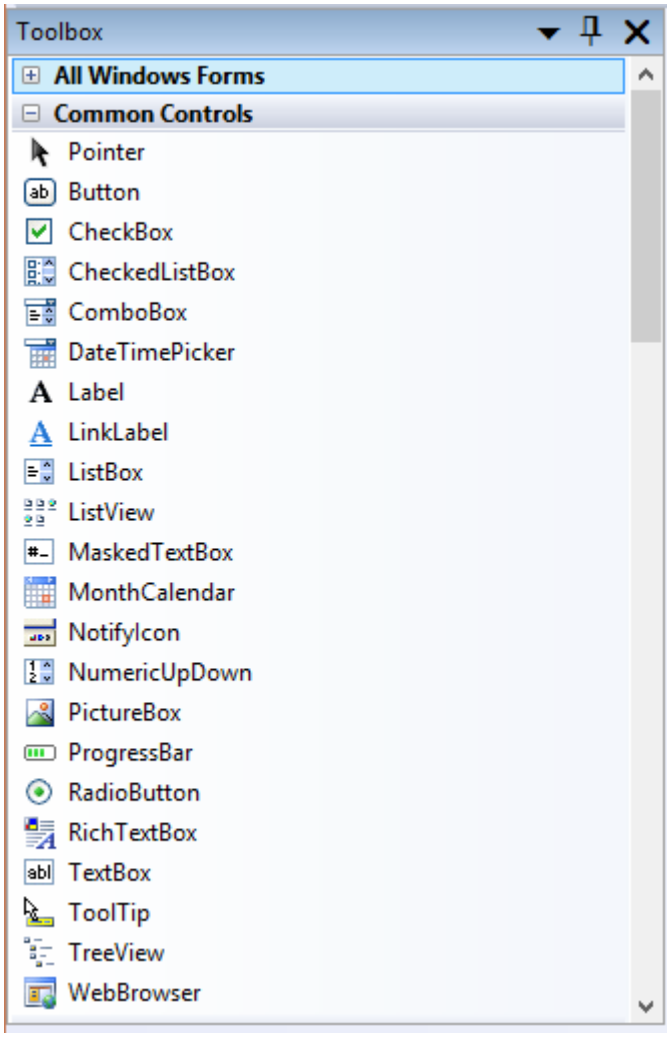
Özellik (Property): Bir nesnenin sahip olduğu değer ya da niteliktir. Tasarım anında özellikler penceresinden(Properties Windows) veya program çalışırken de özellik ayarlaması yapılabilir.

Button: Windows ortamında herhangi bir tuşa basma işlemi için kullanılabilecek durumlarda düşünülebilir. Örneğin; göster, gizle, programdan giriş, çıkış, bitir, son, hesapla vbg. işlemler.

CheckBox: Birden fazla seçeneğin olduğu ve bu birden fazla seçeneğin de aynı anda seçilebildiği durumlarda kullanılabilir. Örneğin; Lokantada onlarca yemek seçeneğinden birkaç tanesini (birden fazlasını) seçebilme durumunda bu nesne kullanılabilir.

RadioButton: Birden fazla seçeneğin olduğu fakat bu birden fazla seçeneğin içinde sadece bir tanesinin seçilebildiği durumlarda kullanılır. Örneğin; "Hangi sınıfta okuyorsunuz?" sorusuna verilebilecek sadece bir tane cevap vardır.

En sık kullanılan nesneler/aletler(ToolBox):



ComboBox: Yerden tasarruf etmek amacıyla, birden fazla seçeneğin açılır bir metin kutusunda toplandığı durumlarda kullanılır. Birden fazla seçeneğin olduğu bütün uygulamalarda kullanılabilir.

ListBox: ComboBox nesnesine benzer, tek farkı seçeneklerin bu nesne içinde bir veya birden fazla satırlık alanda listelenebilmesidir.

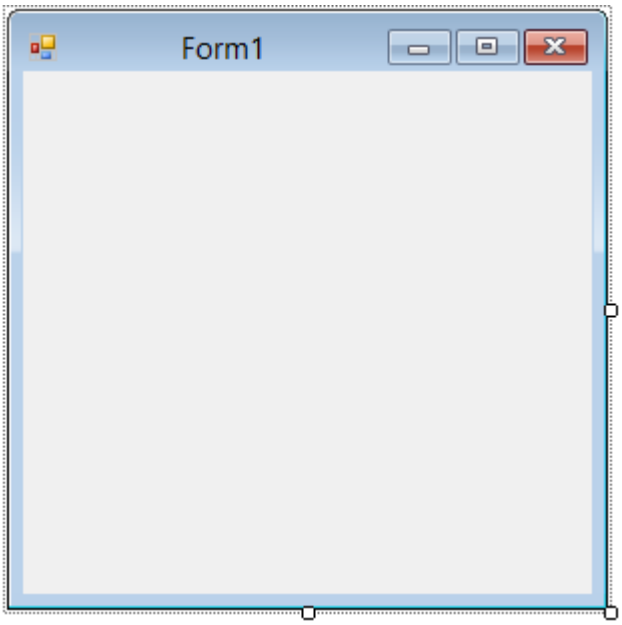
Label: "Sonuç", "İsminiz nedir?", "Sakarya Üniversitesi", "Birinci Sayı" gibi diğer nesneleri tamamlayıcı (tanımlayıcı) ifadeleri form gibi nesneler üzerinde tanımlayabilmek (yazabilmek) için kullanılır.

PictureBox: Form gibi nesneler üzerinde resim gösterebilmek için kullanılır.

TextBox: Metin ifadelerin ekranda gösterilmesi için kullanılır. Örneğin; programın çalışması sırasında elde edilen bir sonucun gösterilmesi, program içinde kullanılacak bilinmeyenlerin sordurulması gibi durumlar için uygundur.

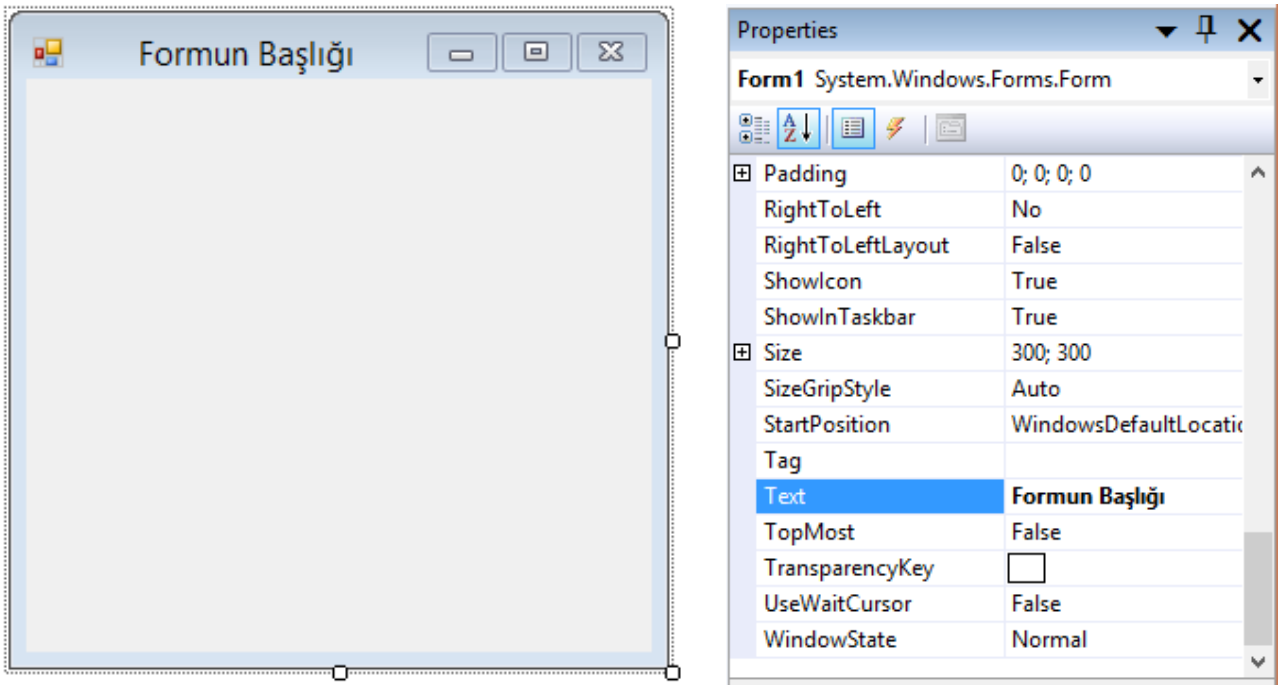
FORMLAR

Visual Basic' te iki çeşit form vardır. Bunlardan bir tanesi tek başına çalışan form türü olan Visual Basic ilk çalıştığı zaman ekrana gelen form ve projeye her eklenen yeni formun türü olan SDI (Single Document Interface) formlarıdır. Bu tür formların içinde başka formlar çalışamazlar. İkinci tür form çeşidi ise içerisinde başka formların çalışmasına izin veren ve MDI (Multi Document Interface) olarak bilinen form türüdür. Bu tür formların içinde Child form olarak adlandırılan ve normal SDI formların Child özelliğinin true yapılmasıyla oluşan formlar çalıştırılabilir.



Formların özellikleri:

Text: Formun başlığına yazılacak yazıyı belirtir. Bu özelliğe formun özellikler penceresinden ulaşılabileceği gibi program içerisinde de formun başlığı değiştirilebilir.



İkon: Formda kullanılacak simgeyi belirtir. Bu simge aynı zamanda programınıza ait bir kısayol oluşturduğunuzda gözükecek simgedir.

BorderStyle : Formun sınırlarının belirlendiği bu özellik aşağıda belirten değerleri alabilir.
0-None: Bu değer verildiğinde form boyutlandırılamaz, taşınamaz ve kapatılamaz. Ayrıca formun çerçevesi, başlığı, kontrol kutusu, min ve max düğmeleri de formda yer almayacaktır.
1-FixedSingle: Bu değer verildiğinde kullanıcı formu büyültüp küçültemez ama formu taşıyabilir ve kapatabilir. Ayrıca formun boyutlarında herhangi bir değişiklik yapılamaz.
2-Sizable: Varsayılan değer budur ve formun tüm özellikleri kullanılabilir.
3-FixedDouble: Kullanıcı formu boyutlandıramaz ama formu taşıyabilir ve formu kapatabilir.
4-FixedToolWindow: Normal forma göre başlığı daha küçük olan ve kontrol menüsü içermeyen bir form oluşturur. Bu formun boyutları kullanıcı tarafından değiştirilemez.
5-SizableToolWindow: Yukarıdaki forma benzer terk farkı boyutları kullanıcı tarafından değiştirilebilir.

MaxButton, MinButton: Formun sağ üst köşesinde bulunan ve formun büyültülüp küçültülmesi işlemleri için kullanılan düğmelerin formda bulunup bulunmamasını belirler.

ControlBox: Formun sol üst köşesinde bulunan kontrol menüsünün görünüp görünmemesini sağlar. Bu değer true veya false değerlerini alabilir. Eğer bu değer true ise kontrol menüsü görünür ve bu form Alt+F4 tuş kombinasyonu kullanılarak kapatılabilir.

Moveable: Bu özellik ile kullanıcının formu taşıyıp taşıyamayacağı belirlenir. Bu özellik true veya false değerini alabilir. Eğer bu değer true ise kullanıcı formu taşıyabilir.

ShowInTaskbar: Bu özellik formun çalışma esnasında görev çubuğunda görünüp görünmeyeceğini belirler. True ya da false değerlerini alabilir. Eğer bu özellik true ise program çalıştığında form görev çubuğunda görünür.

AutoRedraw: Bu özellik ile formun üzerine başka bir form geldiğinde veya formun boyutlarıyla oynandığında formun üzerindeki yazı veya çizimlerin yenilenip yenilenmeyeceği belirlenir. Bu özellik true veya false olmak üzere iki değer alabilir. Eğer bu değer true ise formda yenileme yapılır ve formun üzerindeki yazı ve çizimler kaybolmaz.

FontTransparent: Formun üzerine Print metodu ile yazılan yazıların zemin renginin olup olmamasını belirler. Bu özellik true ya da false olabilir. Eğer true ise yazıların zemin rengi olmayacaktır. False ise yazı kandı zemin rengi üzerine yazılır ve altındaki nesneyi göstermez.

WindowState: Formun ilk çalışmaya başlayacağı zaman alacağı durumu belirler. Bu özellik üç değer alabilir.
0-Normal: Normal durumda açılır.
1-Minimized: Simge durumunda açılır.
2-Maximized: Ekranın tamamını kaplayacak şekilde açılır.

StartupPosition: Formun yüklenmeye başlandığı zaman ekran koordinatlarının neye göre belirleneceğini belirler. Bu özellik dört farklı değer alabilir.
0-Manuel: Form tasarlandığı zamanki konumda açılır.
1-CenterOwner: Child özelliği true yapılmış formların MDI formun ortasında açılmasını sağlar.
2-CenterScreen: Formun ekranın ortasında açılmasını sağlar.
3-WindowsDefault: Formun konumunu Windows belirler.

KeyPreview: Form aktifken basılan tuşlardan formun etkilenip etkilenmeyeceği bu özellik kullanılarak ayarlanılır. Bu özellik iki değer alabilir. Eğer değer true ise formda bulunan herhangi bir kontrolün üzerinde basılan tuşlar ilk önce formun KeyPress, Keydown, KeyUp olaylarını meydana getirir. Bu değer false ise kontrolün üzerinde basılan tuşlar o kontrolün KeyPress, Keydown, KeyUp olaylarını meydana getirir.

Count: Formdaki menüler dahil kontrol sayısını belirtir.

Picture: Formun üzerinde gösterilecek resmi belirtir.

MDIChild: Yukarıda bahsettiğimiz MDI formların içinde çalışabilen bir form istiyorsak bu özelliği true yapmamız gerekiyor.

CurrentX, CurrentY: Formun üzerindeki aktif pixelin yerini belirtir. Formun üzerine yazdırılacak yazılar bu noktadan başlayarak yazdırılır.

Width: Formun genişliğinin belirlendiği özellik.

Heigth: Formun yüksekliğinin belirlendiği özellik.

Left: Formun ekranın ne kadar solunda olacağının belirlendiği özellik.

Top: Formun ekranın ne kadar üstünde yer olacağının belirlendiği özellik.

Font: Formlara yazılacak yazıların fontlarının belirlendiği özellik.

BackColor: Formun zemin renginin belirlendiği özellik.

ScaleMode: Formda kullanılan ölçü birimini belirtir. Varsayılan ölçü birimi Twip'dir.
Kullanılabilecek ölçü birimleri ise;
1-Twip **2-** Point **3-** Pixel **4-** Character **5-** Inch **6-** Millimeter **7-** Centimeter
(1 inch = 1440 twip = 72 point = 2.54 cm)

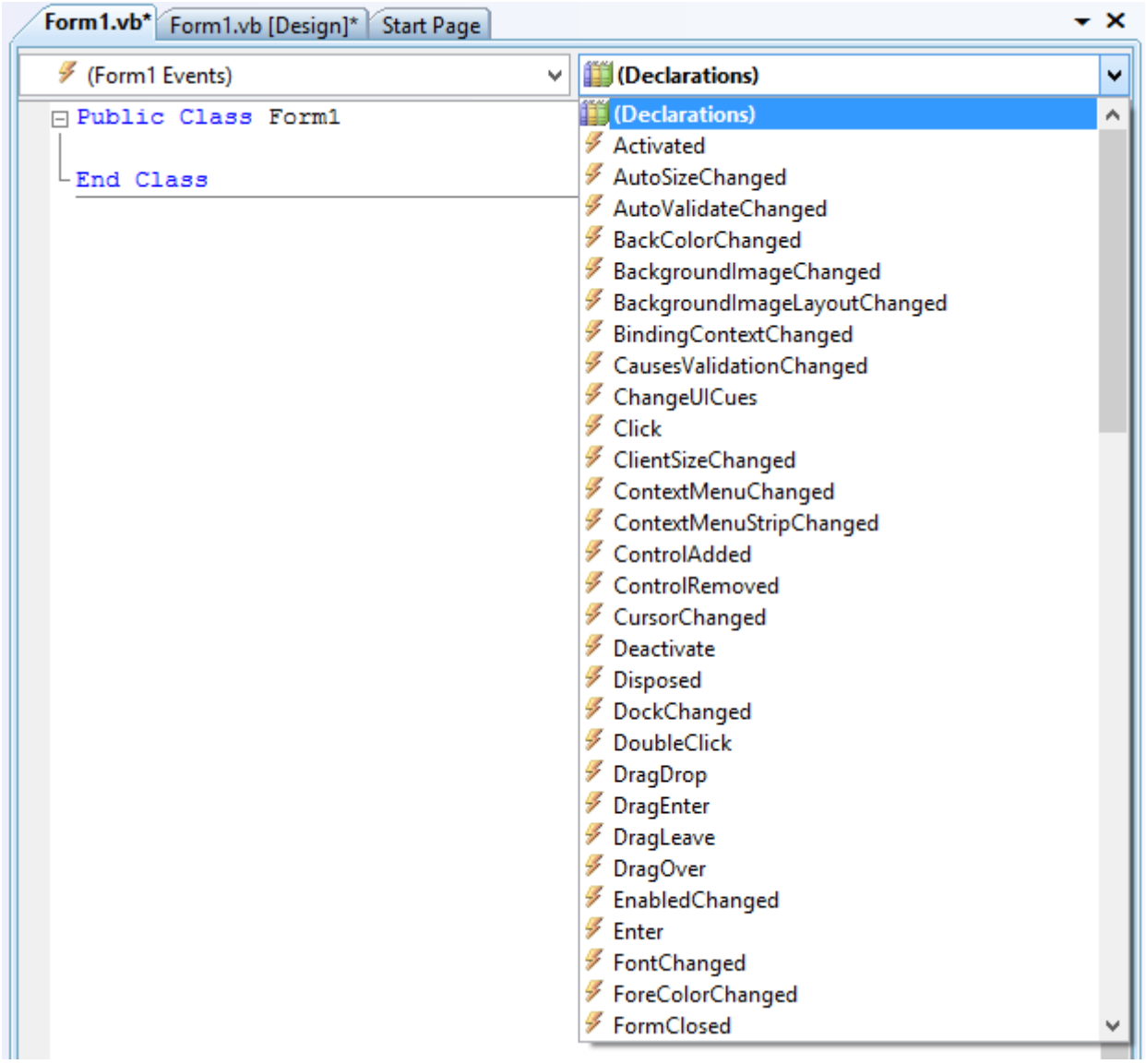
MousePointer: Form üzerindeki fare işaretçisinin şeklini belirler.

Enabled: Formların kullanılabilir olup olmamasının belirlendiği özellik. Eğer bu özellik bir form için false ise o form proje içerisinde kullanılamaz.

VISUAL BASIC'TE FORMLARA UYGULANABİLECEK OLAYLAR

Visual Basic "olay temelli" bir programlama aracı olup, kullanılan kontroller için önceden tanımlanmış olaylara ilişkin kodlar yazılarak programın çalışması sağlanır. Örneğin siz, formun

üzerine çift tıklanması durumunda bir olayın meydana gelmesini istiyorsunuz. Bunun için formda tanımlanmış DoubleClick olayına ilgili kodu yazarak bu olayın meydana gelmesini sağlayabilirsiniz. Şimdi sırasıyla formlara uygulanabilecek olayları inceleyelim.



Load() : Formun ilk defa hafızaya yüklendiği zaman meydana gelen olaydır. Bu olay kullanılarak, kullanıcıya formu göstermeden önce yapılması gereken işler yapılabilir. Örneğin, kullanıcı programı başlattığında ilk önce kullanıcıdan, kullanıcı adı ve şifresini alarak daha sonra yüklenecek ana formda bu bilgileri kullanarak bir veritabanına bağlantı sağlayabiliriz.

Activate() : Formun, programda aktif olması sırasında meydana gelir. Eğer programımızda bir tane form varsa bu form her zaman aktiftir. Bu olaya, formun aktif olması sırasında yapılmasını istediğimiz olayları yazabiliriz.

Deactivate() : Formun, programda aktivitesini kaybetmesi sırasında meydana gelen olaydır. Bu olaya, formun aktivitesini kaybetmesi sırasında yapılmasını istediğimiz olayları yazabiliriz.

Unload(Cancel As Integer) : Formun kapatılması sırasında meydana gelen olaydır. Ayrıca bu olayda tanımlanmış Cancel değişkenini kullanarak sadece bizim istediğimiz durumda formun kapanmasını sağlayabiliriz. Bunu gerçekleştirmek için, yani formun kullanıcı tarafından hiçbir şekilde kapatılmaması için Cancel değişkenine True değerini atamalıyız.

Resize() : Formun boyutlarının değiştirilmesi sırasında meydana gelir. Örneğin formun genişliğinin veya yüksekliğinin değiştirilmesi bu olayın meydana gelmesini sağlar.

KeyPress(KeyAscii As Integer) : Formun çalışması sırasında yön ve kontrol(Ctrl, Alt, Shift) tuşları dışında herhangi bir tuşa basılması sırasında meydana gelen olaydır. Bu olayda basılan tuşun Ascii değeri KeyAscii değişkenine atanır. Bu değişkeni kullanarak hangi tuşa basıldığını öğrenebilirsiniz.

KeyDown(KeyCode As Integer, Shift As Integer) : Bu olay herhangi bir tuşa basıldığı anda meydana gelir. Bu olayda iki parametre kullanılır. Bu parametrelerin biri basılan tuşun Ascii değerini, diğeri ise bu tuşla birlikte Shift, Ctrl ve Alt tuşlarından herhangi birine basılıp basılmadığı konusunda bilgi tutar. Aşağıdaki tabloda Shift parametresinin alabileceği değerler ve açıklamalarını bulabilirsiniz.

Shift	Parametresinin Değeri Açıklama
0	Shift, Ctrl, Alt tuşlarından herhangi birisi basılı durumda değil.
1	Shift tuşu basılı durumda.
2	Ctrl tuşu basılı durumda.
3	Shift ve Ctrl tuşları basılı durumda.
4	Alt tuşu basılı durumda.
5	Shift ve Alt tuşları basılı durumda.
6	Ctrl ve Alt tuşları basılı durumda.
7	Shift, Ctrl ve Alt tuşları basılı durumda.

KeyUp(KeyCode As Integer, Shift As Integer) : Bu olay ise kullanıcının bastığı tuşu bırakması sırasında meydana gelir. Yukarıdaki tabloda Shift parametresi için gösterilen değerler bu olayda da geçerlidir.

MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single) : Bu olay formun üzerinde mouse'un herhangi bir tuşuna basılması sırasında meydana gelir. Bu olayda kullanılan parametrelerden Button parametresi, kullanıcını mouse'un hangi tuşuna bastığı bilgisine tutar ve

kullanıcı mouse'un sol tuşuna basmışsa 1, sağ tuşuna basmışsa 2, her iki tuşa birlikte basmışsa 3 değerini alır. İkinci parametre olan Shift ise mouse'un tuşlarına basıldığı sırada herhangi bir kontrol karakterine basılıp basılmadığı bilgisini tutar ve yukarıdaki tabloda açıklanan değerler bu olaydaki Shift parametresi için de geçerlidir. X ve Y parametreleri ise bu olayının meydana geldiği andaki mouse işaretçisinin yerinin Twip cinsinden değerlerini tutarlar

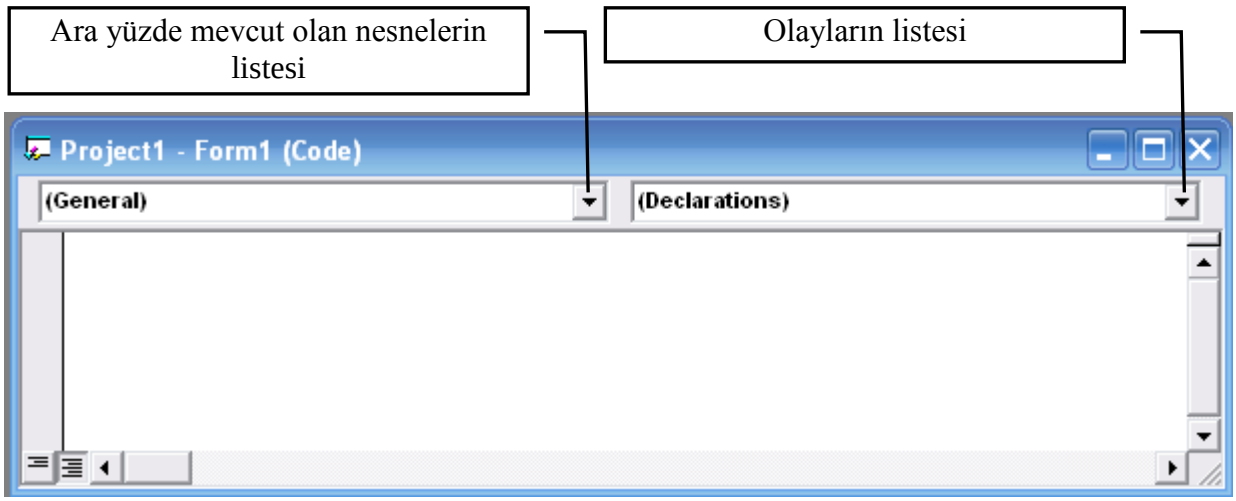
MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single) : Bu olay kullanıcını mouse'un bastığı tuşunu serbest bırakması sırasında meydana gelir ve kullanılan parametreler MouseDown olayındakilerle aynıdır.

MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single) : Bu olay mouse işaretçisinin form üzerindeki yerinin değişmesi sırasında meydana gelir ve kullanılan parametreler MouseDown olayındakilerle aynıdır.

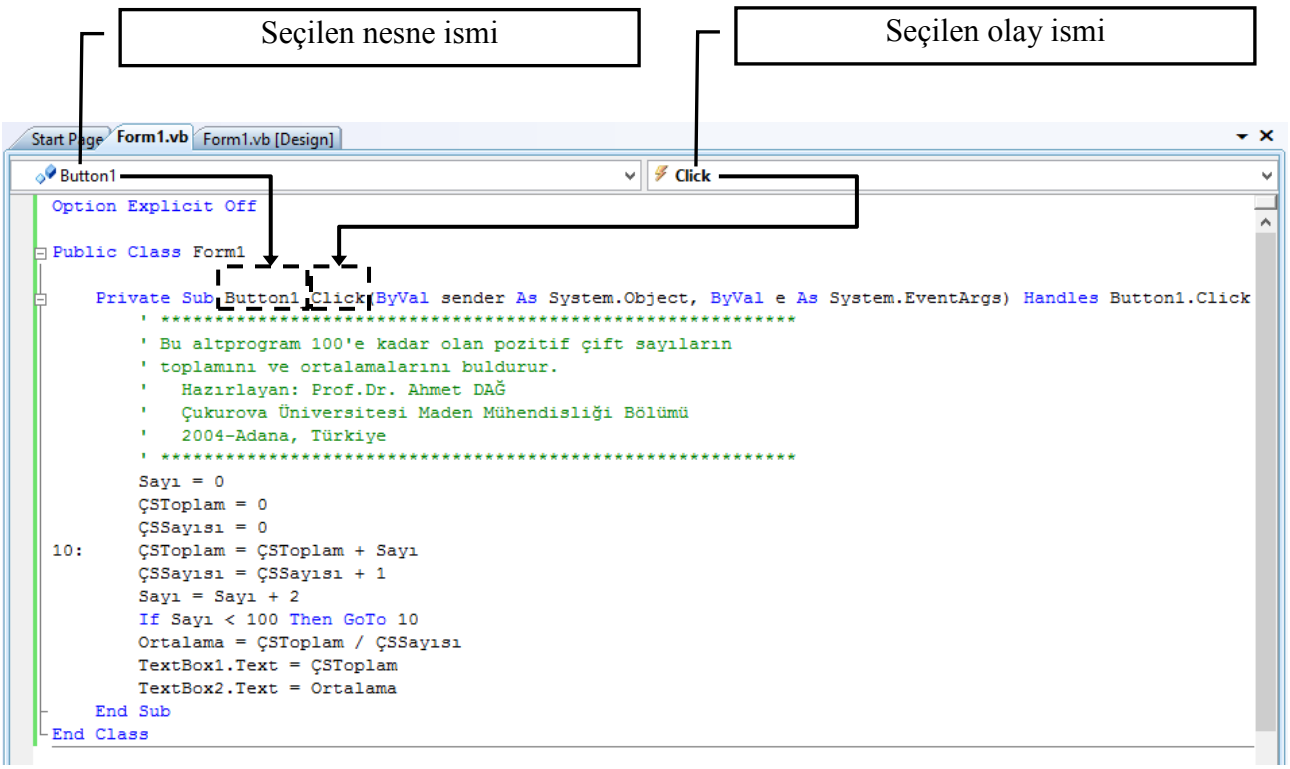
Click() : Bu olay mouse ile formun herhangi bir yerine tek tıklama yapıldığında meydana gelir.

DblClick() : Bu olay mouse ile formun herhangi bir yerine üst üste iki tıklama yapıldığında meydana gelir.

Code Editor: Form üzerinde mevcut olan nesneler ile her bir nesne için oluşturulabilecek olayların(event) listesi görülür. Buradan istenilen bir nesne ve o nesnenin istenilen olayı seçilerek olay gerçekleştiğinde işletilmek istenen kodlar(prosedürler) buraya yazılır.



Aşağıda nesne, olay ve prosedür için örnek bir ekran verilmiştir.



2.1. Değişkenler ve Sabitler

Bütün programlama dillerinin en önemli öğeleri değişkenlerdir. Program içerisinde değerleri sürekli olarak değişebilen yani yeni bir değer atanırsa eski değeri silinen veri yapısıdır. Değişkenler sayesinde verileri hafızaya aktarmak ve veriler üzerinde işlemler yapmak mümkün hale gelmektedir. Program içerisinde değeri değişmeyen veri yapısı sabitlerdir. Verilerin hafızaya aktarılabilmesi için tipleri belirlenmelidir. Örneğin sayısal bir veri hafızaya aktarılırken kullanılacak olan değişkenin veya sabitin veri tipi sayısal olmalıdır. Visual Basic'te alfabetik (karakter), sayısal ve mantıksal olmak üzere üç ana veri tipinin yanında özel veri tipleri de vardır. Bunlara ait bilgiler aşağıdaki Tabloda topluca verilmiştir.

Değişkenler kullanılmadan önce **Dim** deyimiyle, sabitler kullanılmadan önce de **Const** deyimiyle tanımlanması iyi bir alışkanlıktır. Değişken ve sabitlerin önceden tanımlanması sayesinde programların bellekte kapladıkları alanları minimuma indirmek ve bu programların daha hızlı çalışmasını sağlamak mümkündür. Tipi tanımlanmayan değişken ve sabitler variant tip olarak bellekte yer kaplar. Değişkenlerin aksine sabitler bir başlangıç değeriyle tanımlanmak zorundadır. Sabitlerin sahip olduğu değerler değişkenlerde olduğu gibi tekrar değiştirilemezler. Değişken ve sabitlerin tanımlanması ve kullanımında bazı kurallar mevcuttur. Bunlar;

- ✓ Bir harf ile başlamalıdır (A12_MADEN gibi).
- ✓ Harflerin küçük ya da büyük olmasının bir önemi yoktur.
- ✓ Saklanacak olan değeri tanımlayan anlamlı ve en fazla 40 karakter olmalıdır.
- ✓ Özel karakterler içermemelidir (A.B gibi kullanım yanlıştır çünkü "." özel bir karakterdir).

✓ Geçerli bir anahtar sözcük (yasaklanmış isimler) olmamalıdır (**Dim** şeklinde bir değişken ismi tanımlanamaz çünkü "**Dim**" değişken tanımlanması için kullanılan bir anahtar sözcüktür).

✓ Genellikle program birimlerinin (yordamların) başında kullanılırlar.

✓ Değişkenleri tanımlamak için **Dim**, sabitleri tanımlamak için ise **Const** deyimi kullanılır. Aynı türden ve çok elemanlı değerleri içeren bir değişken tanımlaması vardır ki bu değişkenlere indisli değişkenler(array) denir ve özellikle mühendislik problemlerin çözümünde yoğun bir şekilde kullanımı gerekir. İndisli değişken tanımlaması yine **Dim** deyimiyle, boyut ve her boyuttaki eleman sayısı belirtilerek tanımlanır. İndisli değişken tanımlaması yapıldığı zaman aynı değişken ismiyle ve tanımlandığı kadar çok sayıda (indislerin ifade ettiği) değer saklanabilir, bu elemanlara yeni değerler aktarılabilir ve her bir elemanıyla ayrı ayrı işlemler yapılabilir.

Tablo 2. Değişken ve Sabit Tipleri

	Tipi	**Belllek Boyutu**	**Değer Aralığı**
Tamsayı Sayısal	Byte	1 bayt	0 dan 255'e kadar
	Integer	2 bayt	-2147483648 dan 2147483647'e kadar
	Long	4 bayt	-9 223 372 036 854 775 808 dan 9 223 372 036 854 775 807'e kadar
Gerçel Sayısal	Single	4 bayt	$-1.4x10^{45}$ den $3.4x10^{38}$'e kadar
	Double	8 bayt	$-5x10^{324}$ den $1.8x10^{308}$'e kadar
Alfabetik	String	1 bayt	0 dan 2 milyar karaktere kadar
Mantıksal	Boolean	2 bayt	True veya False
Parasal	Currency	8 bayt	$-9x10^{14}$ den $9x10^{14}$'e kadar
Tarih	Date	8 bayt	1/1/100 den 31/12/9999'a kadar
Değişen	Variant	16 bayt+1 bayt	
Nesne	Object	4 bayt	

Değişken Tanımlama:

Değeri değişebilen veya değiştirilebilen değişkenler **Dim** deyimiyle aşağıdaki söz dizimiyle tanımlanabilirler;

Dim *değişkenadı* [**As** *değişken tipi*-] [=başlangıç değeri]

Örnekler;

Dim A **As Double**=3.5 A değişkeni double veri tipinde ve 3.5 başlangıç değeri ile işleme girecektir.

Dim A=5 A değişkeni variant veri tipinde ve 5 (integer-neden integer olduğu yukarıdaki örneklerde anlatıldı) işleme girecektir.

Dim A **As string**=”deneme” A değişkeni string veri tipinde ve “deneme” başlangıç değeri ile işleme girecektir.

Dim A, A1, A2 **As** Double Tek Dim ile çoklu değişken tanımlama

Dim A **As** Double, B **As** Single Tek Dim ile çoklu değişken tanımlama

Değişkenlere başlangıç değeri atanmamışsa;

-Sayı değişken tipleri için (byte, integer, double vbg.) sıfır (O sayısı).

-String değişken tipleri için (char, string vbg.) "" değeridir, bu değerin yazı olarak ifadesi şudur: çift tırnak işareti açılır ve arada boşluk olmadan kapatılır.

-Boolean mantıksal değişken tanımlama tipinde False değeridir.

-Date değişken tanımlama tipinde "00:00:00" değeridir.

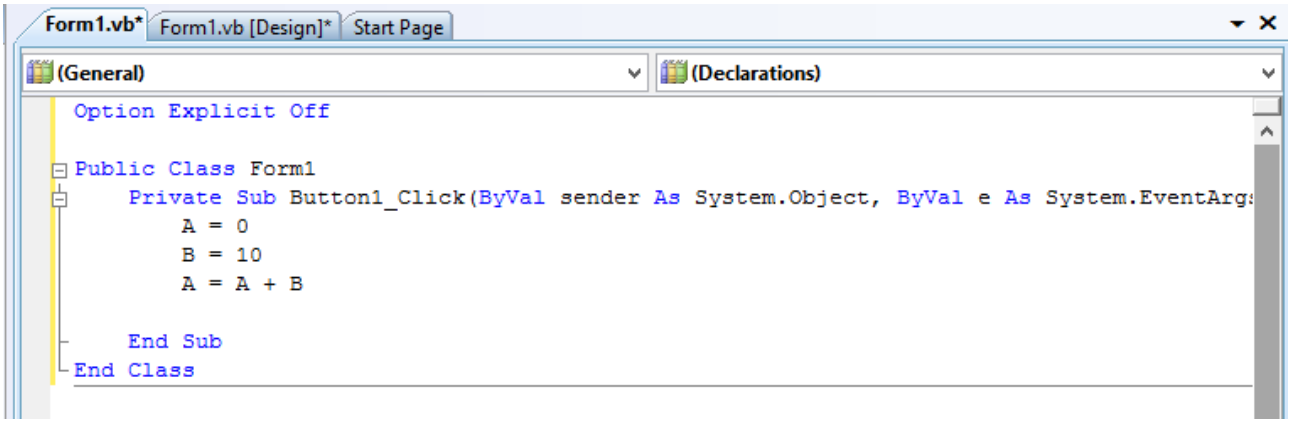
Option Explicit Komutu

Option Explicit komutu, değişkenlerin tanımlanmasını zorunlu hale getirmek veya değişkenleri tanımlama zorunluluğunu ortadan kaldırmak için kullanılır. Bu komutun kullanılma mecburiyeti yoktur, kullanılmazsa, değişkenleri mutlaka tanımlamak zorundasınız demektir (Option Explicit On), aksi durumda, yani değişkenlerin kullanılma zorunluluğu ortadan kaldırılmak istenirse kullanılması mecburidir. Kullanılış şekli;

Option Explicit On (Kullandığınız değişkenleri tanımlamak zorundasınız).

Option Explicit Off (Kullandığınız değişkenleri tanımlamak zorunda değilsiniz)

Option Explicit komutu kullanılmak istenirse, VB editörünün sadece General Declarations bölümüne yazılması gerekir, aksi halde VB hata mesajı verir. General Declarations bölümü; bütün yordamların ve program satırlarının en üstündeki satırdır, aşağıdaki örnekte bu bölümün yeri gösterilmektedir.



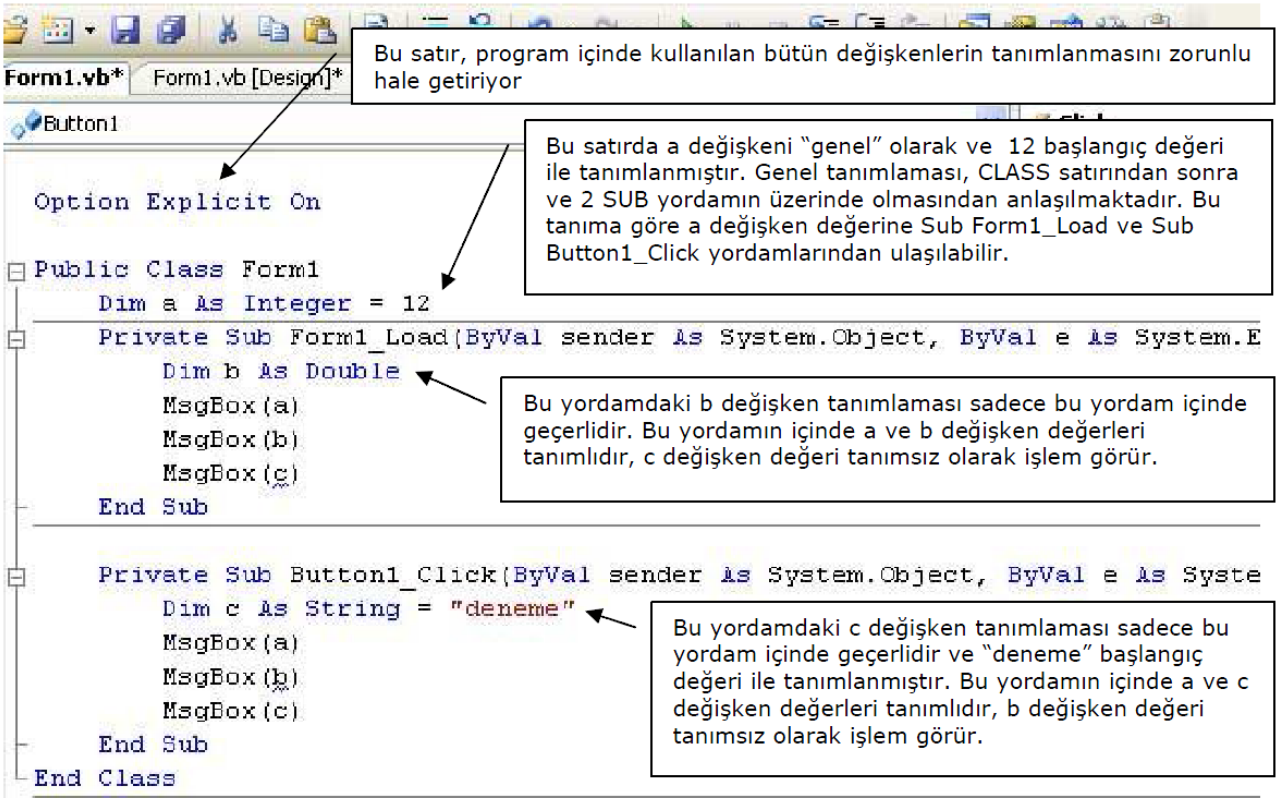
Değişkenlerin tanımlandıklara yere göre durumları:

Değişkenler iki yere göre tanımlanırlar;

Genel tanımlamalar: Yordamların üzerinde yapılan tanımlamadır. Bu şekilde yapılan tanımlar yordamlar üstü olduğu için, Class içindeki bir veya birden fazla yordam içinde bu tanımlar geçerli olacaktır.

Özel tanımlar: Yordamların içinde yapılan tanımlardır. Bu şekilde yapılan tanımlar yordam içinde olduğundan, sadece tanımlandığı yordamlarda geçerli olacaktır, program akışı ilgili yordamın dışına çıktığında, yordam içinde yapılan tanımlar geçersiz olacaktır.

Aşağıdaki örnek, iki tür tanımlamayı da içermektedir. Öncelikle bu programın "Option Explicit On" satırından dolayı çalışmayacağını belirtmek gerekir. Bu programın hatasız çalışması için "Option Explicit On" satırının "Option Explicit Off" haline getirilmesi gerekir. Çünkü bu örnek tanımlamaları göstermek için özellikle bu şekilde düşünülmüştür.




```
Option Explicit On

Public Class Form1
    Dim a As Integer = 12
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.E
        Dim b As Double
        MsgBox(a)
        MsgBox(b)
        MsgBox(c)
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As Syste
        Dim c As String = "deneme"
        MsgBox(a)
        MsgBox(b)
        MsgBox(c)
    End Sub
End Class
```

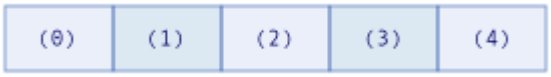
Değişkenlerin statik ve dinamik durumları:

Aksi belirtilmediği sürece tanımladığımız değişkenler dinamiktir; yani olay, prosedürün her çalışmasında (mesela, her command1_click olduğunda) bir önceki işlem sonundaki hesaplanmış değeri saklamazlar. Fakat değişkenimizi tanımlarken Dim X As integer değil de Static X As integer olarak tanımlarsanız her prosedür çalıştığında bir önceki procedure sonundaki hesaplanmış değeri saklar onun üstünden işlem yapar.

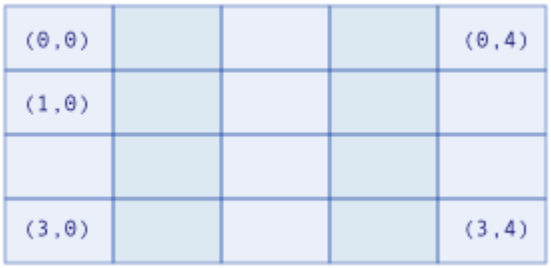
ALT SIRALI (İNDİSLİ) DEĞİŞKENLER (DİZİLER):

Vektör ve matris gibi indislerle ifade edilebilen değişkenler alt sıralı değişken olarak tanımlanır. Böylece aynı isme ve değişken türüne ait bir değerler dizisi elde edilir. Aşağıda farklı boyutlu indisli değişken ve elemanlarına ait şekilsel gösterimler verilmiştir.

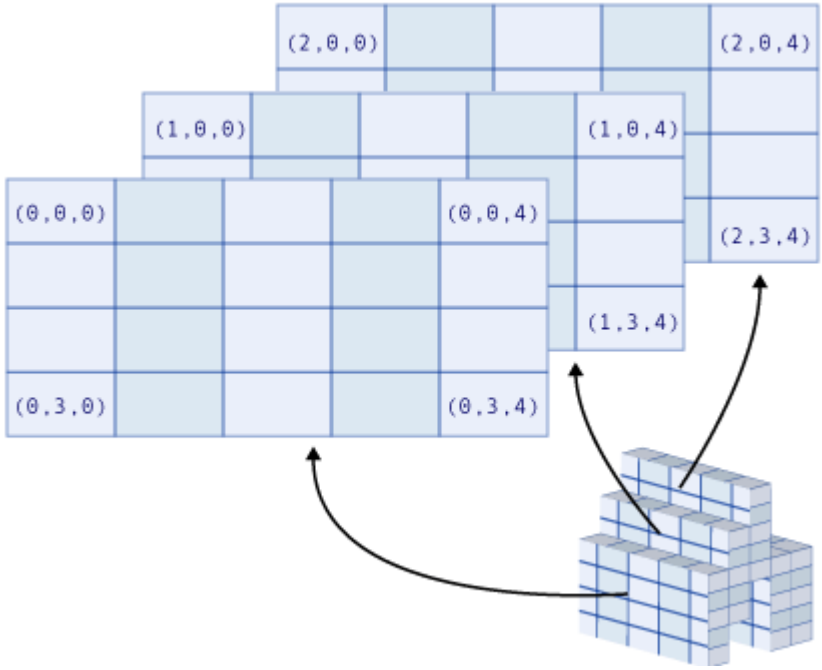
Bir boyutlu dizi(vektör)



İki boyutlu dizi(matris)



Üç boyutlu dizi



ALT SIRALI DEĞİŞKENLERİN TANIMLANMASI

VB'de değişkenler, indislerinin alt ve üst sınır değerleri belirtilerek alt sıralı olarak tanımlanır.

Dim a( )

Dim Vektor(20) **As Integer**

Dim Matris (3, 4) **As Long**

Birinci değişkenin tek boyutlu olup kaç elemanlı olduğu ve değişken türü belirtilmemiştir. Türü Variant olarak alınacak ve kaç elemandan oluşacağı daha sonra **ReDim( )** ile belirtilecektir.

İkinci değişken bir boyutlu, indis değerleri 0 den 20 ye kadar değerler alabilen bir sayı dizisidir ve Integer türünde bir alt sıralı değişkendir. Üçüncü değişken iki boyutludur. Birinci indis 0 den 3 e kadar ikinci indis 0 den 4 e kadar değerler alabilmektedir. (4 x 5) elemanlı bir matris için kullanılabilir. Matrisin elemanları long değişken türünde tanımlanmıştır.

Eğer alt sıralı bir değişkenin indislerinin alt ve üst sınırları belirtilmezse o değişken program içerisinde kullanılmadan önce yeniden tanımlanmalıdır. Buna dinamik alt sıralı değişken denir. İndislerinin sınırları program içinde değişebilen değişkenler için uygundur. Eğer indislerin alt ve üst sınırları belirtilmişse programda bu değerlerin dışına çıkamaz. Böyle değişkenlere statik alt sıralı değişken denir.

```
Dim A( )
Dim B( , )
Dim C( , , )
……
……
i = 10 : j = 100
ReDim A(i)
ReDim B(i, j)
ReDim C(3, i, j)
……
……
```

Burada i ve j nin değerleri program içinde değiştirilerek A değişkeninin sınırları farklı değerler alabilir.

```
' İki boyutlu ve birinci boyutunda (satır) 3 eleman, ikinci boyutunda (sütun) 5 eleman
' olmak üzere tamsayı değerler içeren toplam 3x5=15 elemanlı bir A matrisinin
' tanımlanmasına bir örnek;
```

Dim A**(**0 To 2**,** 0 To 4**) As** Integer veya **Dim** A**(**2**,** 4**) As** Integer

$$A(3,5) = \begin{bmatrix} 10 & 0 & 9 & 1 & -2 \\ 8 & 1 & 11 & 2 & 0 \\ -2 & 2 & 3 & 3 & -1 \end{bmatrix}$$

Yukarıdaki tanımlamaya göre A matrisinin elemanları söyle olur;

A(i,j)	**Açıklama**

(0,0) =	10	A(0,2) + A(2,0) → 7 A(2,4) * A(0,4) → 2 gibi elemanlarıyla istenildiği gibi işlem yapıla bilinir. Ayrıca; A(0,0) = 9 A(1,0) = A(2,1) + A(1,2) gibi elemanlarına yeni değerler aktarıla bilinir.
(0,1) =	0	
(0,2) =	9	
(0,3) =	1	
(0,4) =	-2	
(1,0) =	8	
(1,1) =	1	
(1,2) =	11	
(1,3) =	2	
(1,4) =	0	
(2,0) =	-2	
(2,1) =	2	
(2,2) =	3	
(2,3) =	3	
(2,4) =	-1	

Sabitler **Const** deyimiyle aşağıdaki söz dizimiyle tanımlanabilirler;

Const *sabitismi* [**As** *type*] = *değer*

' Bir sabitin variant tipte ve değer tanımlanmasına bir örnek;

Const Mesaj = "YARDIM"

' Bir sabitin integer tipinde ve değer tanımlamasına bir örnek;

Const Sayı **As** Integer = 5

' Bir satırda birden fazla sabit tanımlanmasına bir örnek.

Const Mesaj As String = "Hello"**,** PiSayısı **As** Double = 3.14159

2.2. Operatörler ve Özel Karakterler

Visual Basic ile programlama sırasında sıkça kullanılan ve gerçekleştirilecek olan işlemlerin temelini oluşturan operatörler bulunmaktadır. Çok sayıda operatör bir arada kullanılırken işlem sırası operatörün öncelik sırasına göre düzenlenir. Parantezler kullanılarak işlem sırası istenildiği şekilde belirlenebilir. Operatörler, matematiksel, karşılaştırma ve mantıksal olmak üzere üç grupta toplanabilirler.

Tablo 3. Matematiksel Operatörler

İşlem	Açıklama	Örnek	Sonuç
+	Toplama	7+2	9
-	Çıkarma	7-2	5
*	Çarpma	7*2	14
/	Bölme	7/2	3.5
\	Tamsayı Bölme	7\2	3
Mod	Kalanlı Bölme	7 Mod 2	1
^	Üs alma	7 ^ 2	49
&	Karakter birleştirme	"7"& "2"	"72"

Tablo 4. Karşılaştırma Operatörleri

İşlem	Açıklama	Örnek	Sonuç
=	Eşittir	7=2	Yanlış(False)
< >	Eşit değildir	7< >2	Doğru(True)
<	Küçüktür	7<2	Yanlış(False)
>	Büyüktür	7>2	Doğru(True)
<=	Küçük veya eşittir(≤)	7<=2	Yanlış(False)
>=	Büyük veya eşittir(≥)	7 >= 2	Doğru(True)

Tablo 5. Mantıksal Operatörler

İşlem	Açıklama
AND	Sadece her iki koşullu ifade Doğru ise sonuç Doğru olur.
OR	Sadece her iki koşullu ifade Yanlış ise sonuç Yanlış olur.
NOT	Koşullu ifade Yanlış ise sonuç Doğru, Doğru ise sonuç Yanlış olur.
XOR	Sadece koşullu ifadelerden biri Doğru ise sonuç Doğru olur.

Tablo 6. Operatörlerin İşlem Öncelik Sıraları

Öncelik	Operatörler
1	Parantezler ( )
2	Matematiksel Operatörler
	2.1. Üs alma(^)
	2.2. Negatifleme(-)
	2.3. Çarpma, Bölme(*, /)
	2.4. Tamsayı bölme(\)
	2.5. Kalanlı bölme(Mod)
	2.6. Toplama, Çıkarma(+,-)
3	Karşılaştırma Operatörleri
	(=, < >, <, >, <=, >=)
4	Mantıksal Operatörler
	4.1. NOT
	4.2. AND
	4.3. OR
	4.4. XOR

Visul Basic de özel anlamı olan değişken veya sabit tanımlamaları yapılırken kullanılmaması gereken özel karakterler de vardır. Bunların bazıları aşağıda belirtilmiştir.

Tablo 7. Bazı Özel Karakterler

Karakter	Açıklama
(	Sol parantez
)	Sağ parantez
.	Nokta veya ondalık kısım belirteci
,	Sabitleri, değişkenleri veya ifadeleri birbirinden ayırır. **Print** deyiminde özel anlamı vardır.
;	Ardışık **Print** deyiminde, değerleri aynı satıra yazmak için kullanılır
:	Aynı satıra yazılmış olan deyimleri ayırır.
?	**Print(=?)** deyimi yerine kullanıla bilinir.
&	İki karakteri birleştirir
'	Üsten tırnak, istenilen yerde açıklama yazmak için kullanılır. Editör de açıklama bilgileri yeşil renkte gözükür.
"	Üsten çift tırnak, karakter değerlerinin sınırlarını belirler.

2.3. Fonksiyonlar

Fonksiyonlar belli işlemleri yaparak bir değer üretirler. Visual Basic'de **Function ( )** deyimiyle tanımlanabilen kullanıcı tanımlı fonksiyonların yanında dilin bir parçası olan hazır (yerleşik) fonksiyonlar da bulunmaktadır. Etkin programlar oluşturabilmek fonksiyonları ve onların

işlemlerini çok iyi bilmekle mümkündür. Hazır fonksiyonlar yaptıkları işlemlere göre belli sınıflara ayrılırlar ve en çok kullanılan hazır fonksiyon sınıfları şunlardır;

- Matematiksel Fonksiyonlar
- String (Karekter) Fonksiyonları
- Dosya Fonksiyonları
- Tarih Fonksiyonları
- Renk Fonksiyonları

Matematiksel hazır fonksiyonlar şu şekilde kullanılır;

Math. Fonksiyon (İfade)

Tablo 8. Bazı Hazır Matematiksel Fonksiyonlar

Math.	Fonksiyon	Kullanımı	İşlevi
$\|x\|$	Abs	Math.Abs(x)	x'in mutlak değerini verir.
arctan(x)	Atan	Math.Atan(x)	tanjantı x olan açının radyan cinsinden değerini verir.
cos(x)	Cos	Math.Cos(x)	x radyan cinsinden olmak üzere cosinüs değerini verir.
sin(x)	Sin	Math.Sin(x)	x radyan cinsinden olmak üzere sinüs değerini verir.
tan(x)	Tan	Math.Tan(x)	x radyan cinsinden olmak üzere tanjant değerini verir.
$\sqrt{x}$	Sqrt	Math.Sqrt(x)	x'in karekök değerini verir.
e^x	Exp	Math.Exp(x)	e sabitinin x kuvvetini verir.
$Log_{10}(x)$	Log	Math.Log(x)	Pozitif bir sayının 10 tabanına göre doğal logaritmasını verir.
	Int	Int(İfade)	Gerçel sayının, değişkenin/math. ifadenin tamsayı kısmını alır.
	Rnd	Rnd[(x)]	0 ile 1 arasında rastgele bir sayı verir.
	Str	Str(x)	x sayısal değerini karaktere çevirir.
	Val	Val(x)	x karakterini sayısal değere çevirir.
Arcsin(x)			Math.Atan(x / Math.Sqr(-x * x + 1))
Arccos(x)			Math.Atan(-x / Math.Sqr(-x * x + 1)) + 2 * Math.Atan(1)
$Log_N(X)$			Math.Log(X) / Math.Log(N)

! ( ) içerisinde bir değişken veya değeri sayısal olan bir matematiksel ifade olabilir

! Açının, dereceden radyana ve radyandan dereceye dönüşüm formülü, $\text{Radyan} = \frac{\text{Derece} \cdot \pi}{180}$

Tablo 9. Bazı Hazır Dosya Fonksiyonları

Hazır Fonk.	**İşlevi**
Bof( )	Dosyanın başını kontrol eder
Eof( )	Dosyanın sonunu kontrol eder
Lof( )	Dosyanın uzunluğunu bayt olarak verir
FileLen( )	Dosyanın uzunluğunu bayt olarak verir
FreeFile( )	Kullanabilir dosya numarasını döndürür

Tablo 10. Bazı Hazır Karakter Fonksiyonları

Hazır Fonk.	**İşlevi**
Len( )	Karekter uzunluğunu bulmak için
Ltrim( )	Solundaki boşluğu kaldırır
Rtrim( )	Sağındakidaki boşluğu kaldırır
Val( )	Karekteri sayısal değere dönüştürür
Str( )	Sayısal değeri karektere dönüştürür.
Mid( )	Ortasından belirtilen değer kadar değer alır
Asc( )	İlk karekterinin ascii koduna dönüştürür
Chr( )	Ascii koduna karşılık gelen karektere dönüştürür
Ucase( )	Büyük karakterlere dönüştürür
Lcase( )	Küçük karakterlere dönüştürür

Tablo 11. Bazı Hazır Zaman Fonksiyonları

Hazır Fonk.	**İşlevi**
Date ( )	Sistem tarih bilgilerini verir
Year ( )	Tarihin yıl değerini verir
Month ( )	Tarihin ay değerini verir
Day ( )	Tarihin gün değerini verir
Time ( )	Sistem zaman bilgilerini verir
Hour ( )	Zamanın saat değerini verir
Minute ( )	Zamanın dakika değerini verir
Second ( )	Zamanın saniye değerini verir

Tablo 12. Bazı Hazır Renk Fonksiyonları

Hazır Fonk.	**İşlevi**
Rgb ( )	(Red, Green, Blue) değerlerinde renk üretir(Her bir renk için 0-255)
QBcolor ( )	(Color) sayısına göre bir renk üretir (0-15)

Asc: Klavyeden girilen herhangi bir ifadenin, nümerik ascii karşılığını verir. Eğer tek harfli ifade yerine sözcük girerseniz, sözcüğün ilk karakterini baz alacaktır. Mesela asc (bilgisayar) denirse b'nin nümerik ascii karşılığı olan 98 elde edilir.

Chr$: 0-255 arasında girilen bir nümerik ascii koduna karşılık gelen karakteri verir.

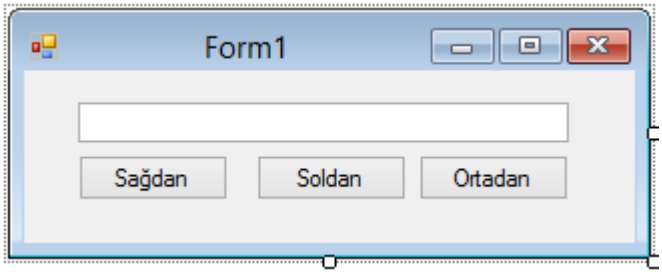
Space$(n): Belirtilen text'e veya herhangi bir yere n kadar boşluk koyar. Text1.text=space$(4) gibi

Right$: Belirtilen stringin en sağından başlayarak belirtilen sayıda stringi kopyalamak için kullanılır. A=Right$("pamukkale",4) derseniz A="kale" olacaktır.

Left$: Belirtilen stringin en solundan başlayarak belirtilen sayıda stringi kopyalamak için kullanılır. A=Leftt$("pamukkale",5) derseniz A="pamuk" olacaktır.

Mid$: Belirtilen stringin belirtilen karakterinden başlayarak, soldan sağa doğru belirtilen kadar karakter kopyalamaya yarar. A=Mid$("pamukkale",4,4) derseniz A="ukka" olacaktır.Yani 4. karakterden başlayarak sağa doğru 4 karakter kopyalayacaktır.

Örnek:



Standart.exe olarak açtığınız form'a 1 tane text kutusu(TextBox1.Text), 3tane buton yerleştirin, butonların adını ister değiştirin ister değiştirmeyin program yinede çalışır. Butonların üzerlerine sıra ile çift tıklayarak sıra ile aşağıdaki kodları yazın. Programı çalıştırdıktan sonra "sağdan" butonuna tıklarsanız, text1 kutusuna ,"pamukkale" stringinin içindeki "kale"yi alıp yazacaktır, "soldan" butonuna tıkladığınızda ise "pamuk", ortadan butonuna tıkladığınızda ise "ukka" yazacaktır...

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        a = Microsoft.VisualBasic.Right("pamukkale", 4)
        TextBox1.Text = a

End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        a = Microsoft.VisualBasic.Left("pamukkale", 5)
        TextBox1.Text = a

End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button3.Click
```

```
        a = Microsoft.VisualBasic.Mid("pamukkale", 4, 4)
        TextBox1.Text = a
End Sub


Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        TextBox1.Text = "pamukkale"
End Sub
```

Len(String): Girilen stringin kaç karakter uzunluğunda olduğunu öğrenmemize yardımcı olur. Sayısal bir değer alır. A=Len("pamukkale") derseniz A=9 olur.

Val(String): Girilen stringi sayısal ifadeye çevirir ve bize, onu toplama, çıkarma gibi işlemlerde kullanma imkanı sağlar. A=val("15") ifadesi text kutusundan veya herhangi bir yolla girilmiş 15 yazısını rakamsallaştırmaya yarar.

Str(Rakam): Val'ın yaptığını tam tersini yapar. Örneğin text kutusunda rakamsal ifadeleri kullanamayız, ancak onları str ile string haline getirip kullanabiliriz. A=Str(15) derseniz A="15" olacaktır.

Lcase(String): İçine girilen küçüklü büyüklü yazıyı tamamen küçük harfe çevirir. A=Lcase("PamUKKale") derseniz A="pamukkale" olacaktır.

Ucase(String): İçine girilen küçüklü büyüklü yazıyı tamamen büyük harfe çevirir. Ucase("PamUKKale") derseniz A="PAMUKKALE" olacaktır.

Instr(Rakam,String1,String2): Birinci karakter içinde ikinci karakteri arar. Eğer aradığını bulursa değeri bulduğu karakterin sıra numarası olur. Örneğin birinci karakterimiz "pamukkale" ikinci karakterimiz "a" yani birinci karakter içinde a'yı arıyacağız. Programı yazıp çalıştırdığımızda değer 2 olacaktır, yani "a" 2. sırada. İsterseniz aramayı istediğiniz sıradan başlatabilirsiniz. Örneğin A=Instr(5,"pamukkale","a") derseniz A=7 olacaktır. Çünkü 5. karakterden aramaya başlattığımız için 7. sıradaki "a" yı gördü.

Trim(String): Parantez içine girilen karakterin sağındaki ve solundaki boşluk karakterini siler.

Ltrim(String): Parantez içine girilen karakterin solundaki boşluk karakterini siler.

Rtrim(String): Parantez içine girilen karakterin sağındaki boşluk karakterini siler.

Date: Bu komut sayesinde sistemin tarihini öğrenebilir yada onu yeniden ayarlayabiliriz. A=Date dersek A o günün tarihi olacaktır. Yalnız atlanılmaması gereken önemli bir ayrıntı ise A yı dim ile tanımlarken "dim A as date" demeliyiz. Eğer bilgisayarın tarihini ayarlamak istiyorsak Date="aa-gg-yyyy". Bir de date olarak tanımladığımız bir değişkene tarih atayabiliriz. Mesela "dim t as date" olarak tanımladığımız değişkene tarih atamak istersek "t=#aa-gg-yyyy#" şeklinde bir ibare kullanmalıyız. Eğer ay kısmına 13 ve daha büyük yada gün kısmına 32 ve daha büyük rakamlar girerseniz programın çalışması esnasında hata verecektir.

Today:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click

TextBox1.Text = DateTime.Today

End Sub
```

ile zamanı bir değişken olarak elimizde tutabiliriz

Now: Programın çalıştığı anki tarih ve zamanı bize verir.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click

TextBox1.Text = DateTime.Now

End Sub
```

VB'DE İŞLETİM SİSTEMİ İLE İLGİLİ KOMUTLAR

MkDir: DOS'tan alınmış bir komuttur. Aktif dizinin altında yada belirlenen yeni bir yola uygun olarak yeni dizin yaratır. Örneğin;

```
MkDir("ders")
```

ile aktif yolun altına yeni bir dizin açabilirsiniz. Bu dizin zaten mevcutsa hata verir.

```
MkDir("C:\Users\User\Desktop\ders")
```

ile istenilen adreste yeni bir dizin açabilirsiniz. Bu dizin zaten mevcutsa hata verir.

RmDir: DOS'tan alınmış bir komuttur. Geçerli yolun altındaki belirtilen klasörü silmeye yarar. Örneğin RmDir "ders" ile aktif yolun altındaki ders dizinini siler. Eğer öyle bir dizin yoksa hata verir. Ayrıca dizinin içi boş olmalı yoksa gene hata verecektir.

Kill: Belirtilen bir dosya veya dosya gurubunu siler. Mesela RmDir ile içi dolu dizini silemezsiniz ilk önce kill "ders*.*" diyerek içini boşaltırsınız. Sonra içi boş dizini RmDir ile silersiniz.

ChDrive: Aktif sürücüyü değiştirir. ChDrive "D" komutu ile aktif sürücüyü D yapmış olursunuz.

ChDir: Aktif dizini değiştirir. Örneğin ChDir "D:\MsOffice\Excel"

Name: Belirtilen dosyanın adını ve yolunu değiştirir. Name "c:\Vb\alan.bas" as c:\Hb\Hesap.bas

CurDir: O anki geçerli olan sürücünün yolunu (aktif dizini) veya o anda kullanılmayan fakat daha önce kullanılmış sürücünün en son yolunu verir. String türü ifade yollar. A=Curdir(c) şeklinde kullanılır.

2.4. Deyimler

2.4.1. Veri Girişi ve Çıkışı

2.4.1.1. Ekrandan Veri Girişi ve Çıkışı

TextBox denetimi (control) ile veri girişi ve çıkışı:

Program çalışırken form üzerinde daha önce dizayn edilmiş (yerleştirilmiş) metin alanlarına kliklenerek karakter olarak bilgi girişi yapıla bildiği gibi hesaplamalar sonucu elde edilen bilgilerin bu alanlara çıkışı da yapıla bilinir. Program çalışırken ekrandan bu denetimle bir değişkene veri girişi yapılmak istenirse şu şekilde tanımlanır.

```
' Text1 araç kutusuna girilen bir karakterin bir karakter değişkene aktarılmasına bir örnek
Adı = TextBox1.Text

' Text1 araç kutusuna girilen bir sayısal değerin bir sayısal değişkene aktarılmasına bir
' örnek
Sayı = Val(TextBox1.Text)
```

Program çalışırken elde edilen bir değerin istenilen text alanına çıkışı yapılmak istenirse şu şekilde tanımlanır.

```
TextBox2.Text = Adı
TextBox3.Text = Sayı
```

InputBox fonsiyonu (function) ile veri girişi:

Bu fonksiyon, program çalışırken istenilen aşamasında kullanıcının klavyeden bilgi girişini sağlamak için kullanılır. Bu fonksiyon işletildiğinde ekrandan giriş için istenilen şekilde bir iletişim kutusu gözükür. Genel kullanımı;

InputBox(Bilgi[, Başlık] [, Varsayılan] [, xpos] [, ypos] [, Helpfile, Context]**)**

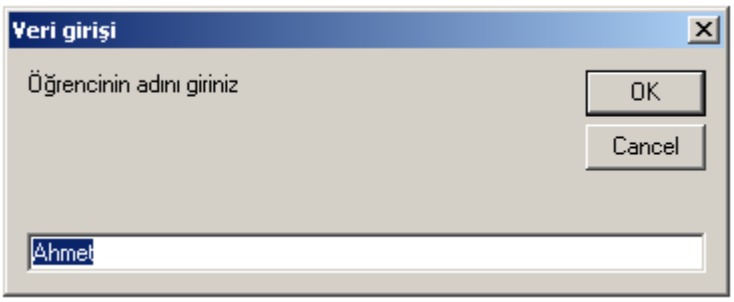
Bilgi: İletişim kutusu üzerinde girilecek olan bilgi hakkında açıklama bilgisi olmalıdır. Bu bilgi, ya daha önce elde edilen bir karakter değişkeni ile ya da " " arasında belirtilen karakter değer ile tanımlanır.

Açıklama: Başlık, Varsayılan, xpos, ypos, Helpfile, Context tanımlamaları isteğe bağlı tanımlamalardır. Bu tanımlamalar fonksiyonun birkaç deneme kullanımıyla anlaşıla bilinir. Bu nedenle burada detaylı açıklamasına gerek duyulmamıştır.

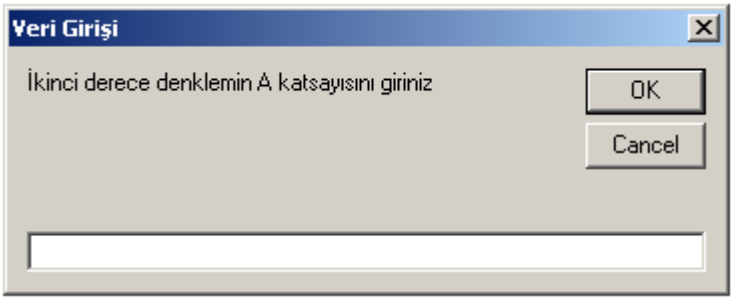
```
' InputBox fonksiyonu ile klavyeden girilecek olan bir karakter değeri  bir karakter
'değişkene aktarılmasına bir örnek
```

Adı = **InputBox**("Öğrencinin adını giriniz", "Veri Girişi","Ahmet")

Yukarıdaki şekliyle tanımlanan bir **InputBox** fonksiyonunun ekranda aşağıdaki iletişim kutusunu oluşturur. Bu haliyle "OK" butonu kliklendiğinde varsayılan değerin girişi yapılabileceği gibi varsayılan olarak "Ahmet" değerinin bulunduğu alana imleç bırakılıp yeni bilgi girişi yapılıp "OK" butonu kliklenerek istenilen değer girilebilir.



' InputBox fonksiyonu ile klavyeden girilecek olan bir sayısal değeri bir sayısal
'değişkene aktarılmasına bir örnek
A = Val(**InputBox**("İkinci derece denklemin A katsayısını giriniz", "Veri Girişi"))



Bunların yanı sıra **ComboBox**, **ListBox** gibi Visual Basic araç kutularıyla da ekrandan bilgi giriş ve çıkışı da yapılabilir. Bu denetimler iler ki konularda değinilecektir.

MsgBox fonsiyonu (function) ile veri çıkışı:

Visual Basic programının çalışması sırasında elde edilen bazı sonuçları ve kullanıcıyı uyaracak gerekli bazı mesajları ekrana taşımamıza yardımcı olur. Kullanımı;

MsgBox("Mesaj"[,Görünüm][,"başlık"]) şeklindedir.

Görünüm ve başlık kısımları kullanılmasa da olur. Bu kısımlar kullanılmazsa sadece MsgBox "mesaj" şeklinde kullanımımız doğru olanıdır. Görünüm kısmında tanımlayacağımız komutlarla ok, cancel, yes, no gibi kontrol butonları koyabileceğiz. Mesela bazen Kullanıcıya ..... yapmak istermisiniz diye sorarsınız, evet mi hayır mı deyip demediğini bu gürünümün değişkene yollayacağı ifadeyle anlarız. Ama görünüm kısmına Hiçbirşey yazmazsak sadece ok butonu

olacaktır mesaj kutumuzda. Görünümde; VbOkOnly, VbOkCancel, VbYesNo, VbYesNoCancel... gibi tanımlamalarla mesaj kutumuza birden fazla buton koyabiliriz. Kullanıcının tıklayacağı buton A değişkenine Vbok, VbCancel... gibi yansıyacaktır. Daha sonra işlemlerimizi bunlara göre yapacaksak, if kontrolüyle bunları yönlendirebiliriz.

Örnek: Standart.exe olarak açtığınız formunuza bir adet text.box koyun ve programın tasarım aşamasındayken forma çift tıklayarak (Private Sub Form_Load() ve End Sub ifadelerini otomatikman elde edebilmek için) aşağıdaki ifadeleri yazınız. Programınızı çalıştırdığınızda isminizi soracak, sonra isminizi text kutusuna yazacak, text kutusunda yazılı olan isminizin doğru olup olmadığını soracak eğer eveti tıkladıysanız program ekranına geri dönecek, hayırı tıkladıysanız bir uyarı mesajı daha gönderecektir.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        a = InputBox("İsminizi Giriniz...")
        TextBox1.Text = a
        Show()
        b = MsgBox("İsminiz Doğrumu?", vbYesNo, "Sorgu Ekranı")
        If b = vbNo Then
            MsgBox("Yanlış Olması İmkansız")
        End If
End Sub
```

2.4.1.2. Dosyadan Veri Girişi ve Çıkışı

Dosyalar (file) büyük miktarda bilgileri bilgisayarın belleğinde depolamak ve bilgisayar da işlemek amacıyla kullanılır. Bir program içerisinde birden fazla dosyadan veri girişi ve/veya çıkış yapılabileceğinden dolayı her bir dosyanın programın başında tanımlanması gerekir.

FileOpen deyimi ile dosya tanımlama:

Bir program içerisinde dosyadan veri girişi ve/veya çıkış yapılabilmesi için programın başında dosyanın **FileOpen** deyimi ile tanımlanması gerekir. Bu deyimin genel kullanımı;

FileOpen(DosyaNumarası, DosyaAdresi, OpenMode [, Access ErişimModu])

DosyaNumarası: Birden fazla dosya ile çalışılabileceği için 1 ile 511 arasında bir tamsayı değer dosya numarası olarak girilmelidir.

DosyaAdresi: Dosyanın adresini ve adını içeren üstten çift tırnak arasında belirtilen karakter ifade ("a:\maden\veriler.dat" veya "veriler.dat" gibi) veya dosya adresine ait karakter değerin saklanıldığı bir karakter değişken (Dosya gibi) olabilir

DosyaModu: OpenMode.**Append**/**Binary**/**Input**/**Output**/ **Random** gibi dosyalara erişim modunu belirten anahtar sözcüklerden birinin girilmesi gerekir (dosya veri girişi için kullanılacaksa **Input**, çıkış için kullanılacaksa **Output** gibi)

ErişimModu: Read, Write veya Read Write anahtar sözcüklerden birinin girilmesi ile açılan dosyaya hangi modda erişime izin verileceğini ifade eden isteğe bağlı bir ayardır

```
' A sürücüsünde var olan bir dosyadan veri girişi erişimi için tanımlanmasına bir örnek;
FileOpen(10, "a:\veriler.dat", OpenMode.Input)

' Adresi veya yolunu ifade eden karakter değişkeni ile veri çıkış erişimi için bir dosya
' tanımlanmasına bir örnek;
Dosya = "c:\belgelerim\program\sonuclar.dat"
FileOpen(2, Dosya, OpenMode.Output)
```

Sıralı Erişimli Dosya Modları:

Append modu: Daha önce oluşturulmuş bir sıralı erişimli dosyanın sonuna yeni kayıt ekleyebilmek için kullanılır. Bilgi girişi bittikten sonra dosya mutlaka Close deyimiyle kapatılmalıdır (Close #DosyaNumarası şeklinde).

Input modu: Daha önce oluşturulmuş ve bilgi girilmiş sıralı erişimli dosyadan bilgi okumak için kullanılır. Bilgi okuma işlemi bittikten sonra dosya mutlaka Close deyimiyle kapatılmalıdır.

Output modu: Sıralı erişimli bir dosya oluşturmak ve dosyaya ilk kez kayıt girmek için kullanılır. Bilgi girişi bittikten sonra dosya mutlaka Close deyimiyle kapatılmalıdır.

Rastgele Erişimli Dosya Modu:

Random modu: Rastgele erişimli bir dosya oluşturmak ve dosyadan aynı anda hem bilgi girme hem de bilgi alma işlemleri yapabilmek için kullanılır. Böyle bir dosyadan bilgi almak için Get deyimi, bilgi kaydı için Put deyimleri kullanılır (Get #DosyaNumarası, KayıtNo, Değişken, veya Put #DosyaNumarası, KayıtNo, Değişken şeklinde). Close deyimiyle kapatılmalıdır.

İkili Dosya Modu:

Binary modu : Hem sıralı hem de rastgele erişilebilen dosya tanımlamak için kullanılır.

Print deyimi ile dosyaya veri çıkışı(Kaydı-Yazma):

Open deyimiyle sıralı erişimli Output modda tanımlanan bir dosyaya tanımlanan değerlerin veya değişkenlerin değerlerinin istenilen formatta yazılmasında kullanılır. Genel kullanımı;

Print(DosyaNumarası, [İfadeler])

DosyaNumarası: Geçerli bir dosyanın numarası.

İfadeler: tanımlanması isteğe bağlıdır. Eğer tanımlanacaksa, bir veya birden fazla virgül veya noktalı virgül ile tanımlanmış, değeri dosyaya yazılacak olan ifadeler(değer, değişken, matematiksel veya mantıksal ifadeler) olmalıdır.

```
' Bir kullanım örneği;
FileOpen (1, "Deneme" OpenMode.Output)   ' çıkış için bir dosya açar.
Print (1, "Maden Mühendisliği Bölümü", 10)   ' değerleri virgül ile dosyaya yazar.
Print (1)   ' boş bir satır yazar.
FileClose (1)   ' dosyayı kapatır.
```

PrintLine deyimi ile dosyaya veri çıkışı(Kaydı-Yazma):

Open deyimiyle sıralı erişimli Output modda tanımlanan bir dosyaya tanımlanan değerlerin veya değişkenlerin değerlerinin satır satır yazılmasında kullanılır. Genel kullanımı;

PrintLine(DosyaNumarası, [İfadeler])

Input deyimi ile veri girişi(okuma):Open deyimiyle sıralı erişimli **Input** modda tanımlanan bir dosyadan veri okur ve okunan verileri tanımlanan değişkenlere aktarır. Genel kullanımı;

Input(DosyaNumarası, Değişkenler)

DosyaNumarası: Veri girişi için erişim sağlanmak üzere Open deyimiyle tanımlanmış olan dosyanın numarası.

Değişkenler: Bir değişken veya virgüllerle ayrılmış değişkenler tanımlanmalıdır. Dosyadan sıralı olarak erişilen aktarılacak bilgilerin türlerine(sayısal, karakter veya mantıksal gibi) uygun türde değişkenlerin tanımlanması gerekmektedir. Erişilen bilgi sayısal bir değer ise ve bu değerin aktarılacağı değişken türü sayısal türde değilse böle bir durumda okuma hatası oluşur.

```
' Veri girişi için sıralı erişim dosya tanımlaması ve bu dosyadan veri giriş tanımlamasına
' bir örnek;

FileOpen(2, "veriler", OpenMode. Input)
Input (2, Ad_Soyad, Not)
```

LineInput deyimi ile veri girişi(okuma):Open deyimiyle sıralı erişimli **Input** modda tanımlanan bir dosyadan satır satır veri okur ve okunan verileri tanımlanan değişkenlere aktarır. Genel kullanımı;

LineInput(DosyaNumarası, Değişkenler)

Örnek;

```
FileOpen(2, "veriler", OpenMode. Input)
Ad_Soyad = LineInput (2)
```

2.4.2. Atama (Değer aktarımı) Deyimi (=)

Bu deyim ile = ‘in sol tarafında belirtilen değişkene hafızada veri girişi veya hesaplamalar sonucu elde edilen değerlerin aktarımında kullanılır. Genel kullanımı;

Değişken = Değer [veya değişken veya bir ifade]

```
' Bir mantıksal değişkene bir değer aktarımı;
Pdeger = True  ' Değişken hafızada True değerini alır
' Bir mantıksal değişkene mantıksal bir ifadeyle bir değer aktarımı;
Qdeger = Pdeger  And 10^2 < > A      ' Değişken ifadenin sonucuna göre True veya False değerini alır

' Bir sayısal değişkene bir değer aktarımı;
Toplam = 10.1       ' Değişken hafızada 10.1 değerini alır.
' Bir sayısal değişkene matematiksel bir ifadeyle değer aktarımı;
Toplam = Toplam + A – 10 + Exp(1-Sqr(A))

' Bir karakter değişkene bir değer aktarımı;
Adı = "Ahmet" : Soyad = "Dağ"


' Bir karakter değişkene bir ifade ile değer aktarımı;
Adı = "Ahmet " & Soyad
```

2.4.3. Gönderme Deyimleri

GoTo Deyimi:

Yordam içerisinde belirtilen satıra şartsız olarak gönderimi (dallanmayı) sağlar. Sözdizimi;

GoTo SatırNumarası

Satır satır işletilecek olan Deyimlere tek olmak kaydıyla ve ilk kolondan başlamak üzere bir satır numarası verilebilir. Her satıra bir satır numarası verme zorunluluğu yoktur. Genel de bir karar deyimiyle birlikte de kullanılabilir. Tanımlanan SatırNumarası yordam içerisinde var olan bir satır olmalıdır.

```
' Koşula göre uygun satıra gönderime bir örnek;
If  Sayı  = 1 Then GoTo 30 Else GoTo 10
If  Sayı  = 1 Then GoTo 30
```

2.4.4. Karar (Denetim) Deyimleri

Programlarda farklı durumlarda farklı işlemlerin yapılması istenir. Bu gibi durumlarda koşul veya duruma göre programın işleyiş akışını denetleyen karar deyimleri kullanılır. Karar deyimleri duruma göre yapılacak işlemlerin seçimini sağlarlar. Visual Basic'de iki türlü karar yapısı vardır. Bunlar; If ..... Then ..... ve Select Case deyimleridir.

2.4.4.1. If ..... Then .... Else Deyimi

Bir koşula göre yapılacak olan işlemlerin işletilmesini sağlar. Genel kullanımı;

If Koşul **Then** [Deyimler(işlemler)] [**Else** Deyimler]

veya aşağıdaki şekilde blok formda da kullanılır.

If Koşul1 **Then**
[Deyimler1]
[**ElseIf** Koşul2 **Then**
[Deyimler2]]
[**ElseIf** Koşul3 **Then**
[Deyimler3]]
-
-
[**ElseIf** KoşulN **Then**
[DeyimlerN]]
[**Else**
[Deyimler]]
End If

Koşul: Mantıksal bir değeri olan ifade olmalıdır. Koşul doğru ise tanımlanan işlemleri gerçekleştirir ve karar deyiminden sonra devam eder.

Örnek; Aşağıdaki şekilde tanımlanan bir F(x) fonksiyonun ekrandan (TextBox denetimiyle) girilen x değerlerine göre F(x)'in değerini hesaplayıp sonucu ekran yazdıran bir programı If .... Then karar yapısı kullanılarak söyle oluşturulur;

$$F(x)=\begin{cases}\sqrt{|x|} & , \quad x<0\\ -10 & , \quad x=0\\ \dfrac{e^{x}+20}{x+2} & , \quad x=1\\ \mathrm{Log}_2(x)+\mathrm{Log}_{10}(x+5) & , \quad x=2\\ \mathrm{Sin}(x)-\mathrm{Tan}(\dfrac{x}{\pi}) & , \quad x=3\\ \dfrac{100-x3}{x+2}-20 & , \quad x>3\end{cases}$$

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        FileOpen(10, "C:\Users\User\Desktop\deneme",
OpenMode.Output)
        Dim x As Long, F As Double
        Const Pisabiti As Double = 3.14159265358979
        x = Val(TextBox1.Text)
        If x < 0 Then
            F = Math.Sqrt(Math.Abs(x))
        ElseIf x = 0 Then
            F = -10
        ElseIf x = 1 Then
            F = (Math.Exp(x) + 20) / (x + 2)
        ElseIf x = 2 Then
            F = Math.Log(x) / Math.Log(2) + Math.Log(x + 5)
        ElseIf x = 3 Then
            F = Math.Sin(x) - Math.Tan(x / Pisabiti)
        Else
            F = (100 - x ^ 3) / (x + 2) - 20
        End If
        Print(10, "F(", x, ")=", F)
        MsgBox("İşlem bitti!")
        FileClose(10)
End Sub
```

2.4.4.2. Select Case Deyimi

If ..... Then karar deyimine benzer olarak bir koşul ya da değişkenin olası değerlerini test ederek bu test değerine göre istenilen işlemlerin işletilmesini sağlar. Genel kullanımı;

Select Case TestİfadesiBlok

Case Değerler1

İşlemler1

[**Case** Değerler2

[İşlemler2]]

-

-

[**Case** DeğerlerN

[İşlemlerN]]

[**Case Else**

[İşlemlerN+1]]

End Select

TestİfadesiBlok: Değeri sayısal olan bir ifade olmalıdır. İfadenin sayısal değerlerine göre istenilen gerçekleştirir ve End Select deyiminden sonra devam eder.

Değerler: Gerçekleştirilecek olan işlemlerin ifadenin hangi değer veya değerlerinde yapılacağını tanımlar. Bu tanımlama iki değer aralığını tanımlamak için To anahtar sözcüğü ile tanımlanır. Eğer değer aralığı mantıksal karşılaştırma operatörü kullanılacak ise Is anahtar sözcüğü ile tanımlanır. Birden fazla değer veya değer aralığı virgül ile tanımlanabilir.

Örnek; Aşağıdaki şekilde tanımlanan bir F(x) fonksiyonun ekrandan (TextBox denetimiyle) girilen x değerlerine göre F(x)'in değerini hesaplayıp sonucu ekran yazdıran bir program Select Case deyimi kullanılarak söyle oluşturulur;

$$F(x)=\begin{cases}\sqrt{|x|} & , \quad x<0\\ -10 & , \quad x=0 \text{ veya } x=5\\ \dfrac{e^{x}+20}{x+2} & , \quad 0\langle x\langle 3\\ Log_2(x)+Log_{10}(x+5) & , \quad x=3\\ Sin(x)-Tan(\dfrac{x}{\pi}) & , \quad x=6\\ \dfrac{100-x3}{x+2}-20 & , \quad x=4 \text{ veya } x>6\end{cases}$$

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        FileOpen(10, "C:\Users\User\Desktop\deneme",
OpenMode.Output)
        Dim x As Long, F As Double
        Const Pisabiti As Double = 3.14159265358979
        x = Val(TextBox1.Text)
        Select Case x
            Case Is < 0
                F = Math.Sqrt(Math.Abs(x))
            Case 0, 5
                F = -10
            Case 1 To 2
                F = (Math.Exp(x) + 20) / (x + 2)
            Case 3
                F = Math.Log(x) / Math.Log(2) + Math.Log(x + 5)
            Case 6
                F = Math.Sin(x) - Math.Tan(x / Pisabiti)
            Case 4, Is > 6
                F = (100 - x ^ 3) / (x + 2) - 20
        End Select
        Print(10, "F(", x, ")=", F)
        MsgBox("İşlem bitti!")
        FileClose(10)
End Sub
```

2.4.5. Döngü Deyimleri

Program denetiminde yaygın olarak kullanılan deyimlerden birisi de döngülerdir. Döngüler yapılacak olan işlemlerin yinelenmesi(tekrar edilmesi) anlamına gelir. Örneğin kullanıcıdan çok

sayıda değer girişini sağlamak için döngüler kullanılabilir. Visual Basic'de, sayaçlı ve koşullu olmak üzere iki tip döngü yapısı vardır.

2.4.5.1. Sayaçlı Döngü

Bu döngü ile işlem veya işlemlerin istenilen sayıda işletilmesini sağlarlar. Bu şekilde bir döngü tanımlaması For... Next deyimiyle yapılır. Bu deyimin genel kullanımı;

For DöngüBelirteci = DöngüBaşlangıçı **To** DöngüSonu [**Step** Artırım]
İşlemler1
[**Exit For**]
İşlemler2
Next [DöngüBelirteci]

veya

For DöngüBelirteci = DöngüSonu **DownTo** DöngüBaşlangıçı [**Step** Azaltım]
İşlemler1
[**Exit For**]
İşlemler2
Next [DöngüBelirteci]

DöngüBelirteci: Sayısal bir değişken adı olmalı.
DöngüBaşlangıçı: Sayısal bir değer, sayısal bir değişken veya değeri sayısal olan bir ifade olabilir.
DöngüSonu: Sayısal bir değer, sayısal bir değişken veya değeri sayısal olan bir ifade olabilir.
Arttırım/Azaltım: Sayısal bir değer, sayısal bir değişken veya değeri sayısal olan bir ifade olabilir. Bu değer kullanılmaz ise arttırım veya azaltımın 1 olarak varsayıldığı kabul edilir.
Açıklama: Döngü belirteci ilk değer olarak tanımlanan başlangıç değerini next'e kadar olan deyimleri işletir ve döngü başına döner sonra döngü belirteci artırım veya azaltım kadar artarak veya azalarak değer alır bu değer döngü sonunu belirten değerle kıyaslanır. Eğer döngü sonundan büyük ise döngüden çıkar değil ise istenilen deyimleri tekrar işleterek aynı işlemleri tekrar eder.

Örnek: Ekrandan girilen N adet sayının toplamını bulan bir program yapılmak istenirse for next döngüsü ile şöyle olur;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
```

```
        FileOpen(10, "C:\Users\User\Desktop\deneme",
OpenMode.Output)
        N = Val(InputBox("Girilecek olan değerlerin sayısını(N)
giriniz"))
        For i = 1 To N Step 1
            Sayı = Val(InputBox("Sayıyı giriniz"))
            Toplam = Toplam + Sayı
        Next i
        Print(10, "Girilen", N, "adet sayının toplamı=", Toplam)
        FileClose(10)
        MsgBox("İşlem bitti!")
End Sub
```

2.4.5.2. Koşullu Döngü

Bu döngü, belirli bir işlem olduğu sürece veya koşul sağlanıncaya kadar istenilen işlemlerin işletilmesini sağlar. **Do Loop** koşullu döngü yapısında döngü başlangıcını **Do** sonunu ise **Loop** deyimleri belirtir. Döngünün başında veya sonunda **While** veya **Until** deyimleri ile birlikte kullanılır. **While** deyimi kullanılırsa, belirtilen koşul doğru olduğu sürece döngü işletilir. **Until** deyimi kullanılır ise belirtilen koşul doğru oluncaya kadar döngü işletilir. Kullanımı;

Do [{**While** | **Until**} Koşul]
[İşlemler1]
[**Exit Do**]
[işlemler2]
Loop

veya

Do
[İşlemler1]
[**Exit Do**]
[İşlemler2]
Loop [{**While** | **Until**} Koşul]

Koşul: Değeri doğru veya yanlış olan bir mantısal ifade olmalıdır.

Örnek; C sürücüsünde var olan "veri.txt" dosyası içerisinde mevcut olan değerlerin sayısını bulduran bir programa.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayac = 0
```

```
        FileOpen(10, "C:\Users\User\Desktop\deneme",
OpenMode.Output)
        FileOpen(1, "c:\Users\User\Desktop\veri.txt",
OpenMode.Input)
        Do While Not EOF(1)
            Input(1, Deger)
            Sayac = Sayac + 1
        Loop
        Print(10, Sayac)
        FileClose(1)
        FileClose(10)
        MsgBox("İşlem bitti!")
End Sub
```

Veya

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayac = 0
        FileOpen(10, "C:\Users\User\Desktop\deneme",
OpenMode.Output)
        FileOpen(1, "c:\Users\User\Desktop\veri.txt",
OpenMode.Input)
        Do
            Input(1, Deger)
            Sayac = Sayac + 1
        Loop Until EOF(1)
        Print(10, Sayac)
        FileClose(1)
        FileClose(10)
        MsgBox("İşlem bitti!")
End Sub
```

2.5. Fonksiyon (Function) ve Altprogram (Subroutine) Oluşturma

Daha önce belirtilen hazır fonksiyonlara ek olarak programcı istenilen işlemler sonucunda tek değer üreten ve istenildiğinde kullanılmak üzere kullanıcı tanımlı fonksiyonlar oluşturup kendi kütüphanelerine kaydedebilirler. Ayrıca istenilen işletimleri gerçekleştiren altprogramlar(Sub) da oluşturulup istenildiğinde işletilebilir. **Sub** geriye bir değer döndürmez iken Function geriye

mutlaka bir değer döndürür. Hazırlanacak **Function** veya **Sub** sadece geçerli formda kullanılabilir olması isteniyorsa **Private**, tüm projede kullanılacak ise **Public** olarak tanımlanır. Belirtilmez ise varsayılan yapı Private olacaktır.

Alt program oluşturma;

[Private veya **Public][Static]Sub** *altprogramismi* ([*değişkenler*])
deyimler
End Sub

Altprogram çağırma;

Call *altprogramismi* ([*ifadeler*])
veya
altprogramismi yapaydeğişken1, yapaydeğişken2, yapaydeğişken1, …

Fonksiyon oluşturma;

[**Public** veya **Private**] [**Static**] **Function** *fonksiyonismi* **(***değişkenler***)** [**As** *tipi*]
[*deyimler*]
[*fonksiyonismi* = *ifade*]
[**Exit Function**]
[*deyimler*]
[*fonksiyonismi* = *ifade*]
End Function

Fonksiyon çağırma;

Call *fonksiyonismi* **(***ifadeler***)**
veya
Herhangi bir atamada veya ifadede *fonksiyonismi* **(***ifadeler***)** şeklinde

Sub tanımlama ve çağırmaya örnek;

```
Public Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        c = 10
        d = 20
```

```
        Call Topla(c, d)
End Sub
    Public Sub Topla(ByVal a, ByVal b)
        sonuc = a + b
        TextBox1.Text = sonuc / 2 'Fonksiyona göre Topla
değişkenine c ve d'nin toplamı olan 30 değerini üretir ve 2'ye
bölerek ekrana 15 değerini yazar
    End Sub
```

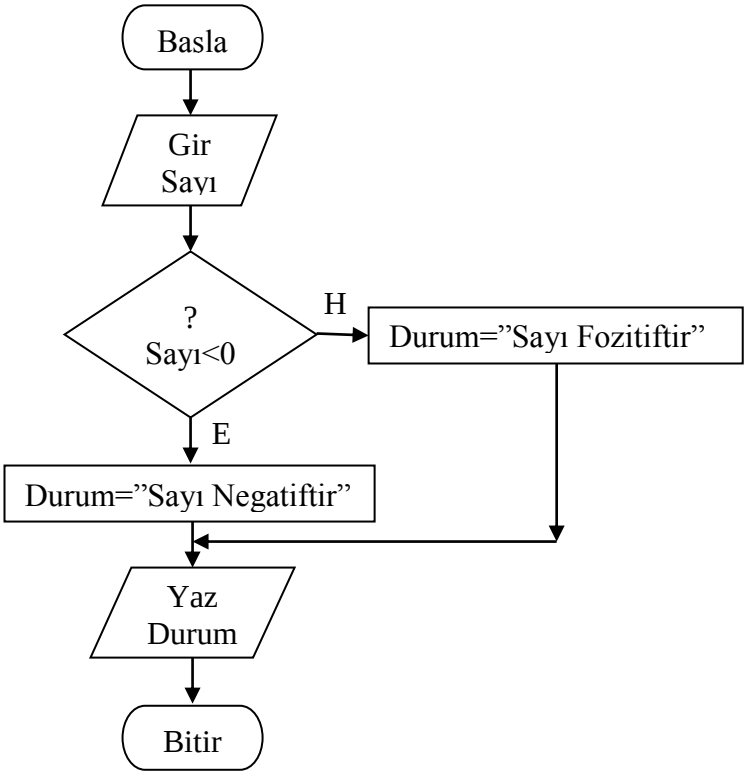
Function tanımlama ve çağırmaya örnek;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        c = 10
        d = 20
        TextBox1.Text = Topla(c, d) / 2 'Fonksiyona göre Topla
değişkenine c ve d'nin toplamı olan 30 değerini üretir ve 2'ye
bölerek ekrana 15 değerini yazar
End Sub
    Public Function Topla(ByVal a, ByVal b)
        Topla = a + b
    End Function
```

3. BAZI PROBLEMLERİN ALGORİTMASI

3.1. Girilen bir sayının pozitif veya negatif olduğunun belirlenmesi

Programın Akış Diyagramı;

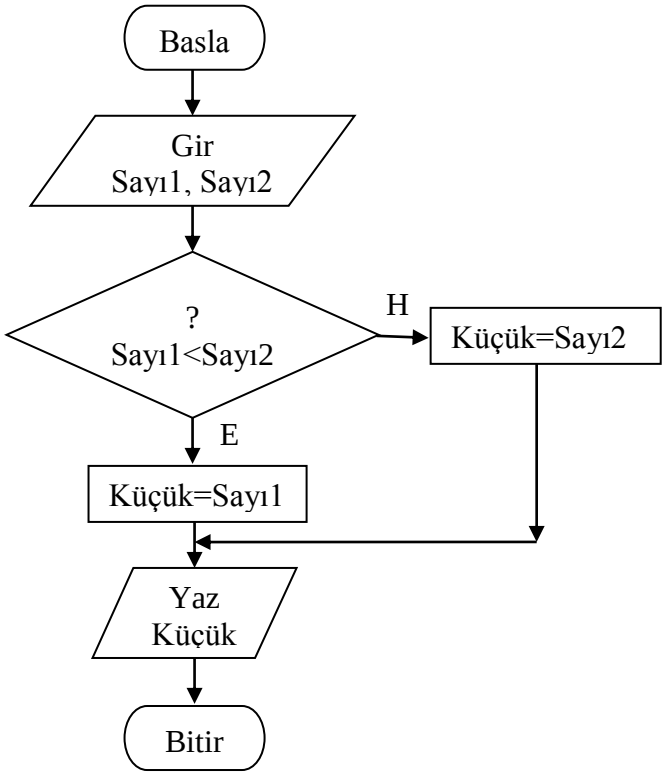


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayı = Val(InputBox("Sayıyı giriniz"))
        If Sayı < 0 Then
            Durum = "Sayı Negatiftir"
        Else
            Durum = "Sayı Pozitiftir"
        End If
        MsgBox(Durum)
End Sub
```

3.2. Girilen iki sayıdan büyük/küçük olanın belirlenmesi

Programın Akış Diyagramı;

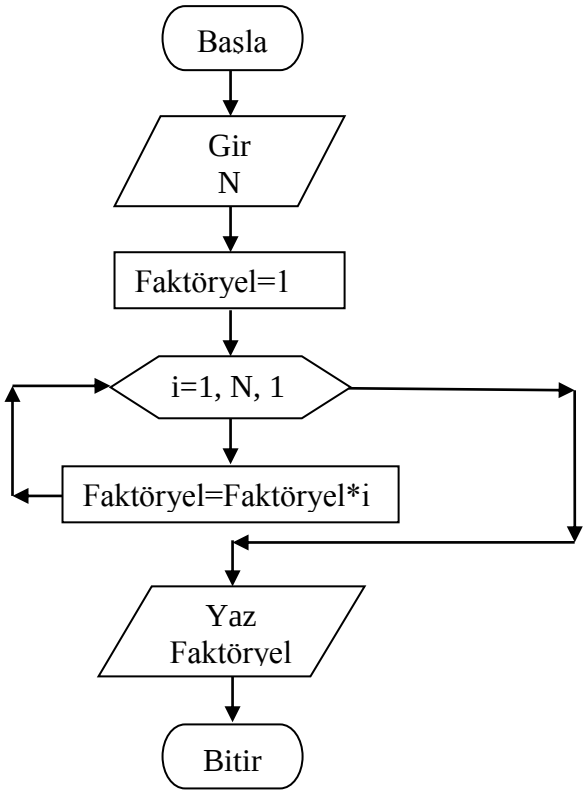


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayı1 = Val(InputBox("Birinci sayıyı giriniz"))
        Sayı2 = Val(InputBox("İkinci sayıyı giriniz"))
        If Sayı1 < Sayı2 Then
            Küçük = Sayı1
        Else
            Küçük = Sayı2
        End If
        MsgBox("Küçük olan sayı " & Küçük & " dir.")
End Sub
```

3.3. Girilen bir sayının faktöriyelini hesaplama(Faktöryel=N!=1*2*….*N)

Programın Akış Diyagramı;

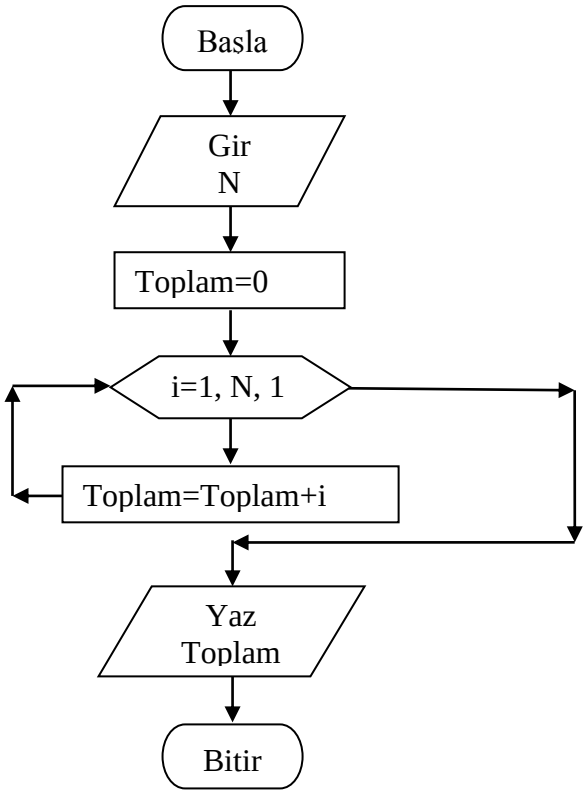


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        N = Val(InputBox("Faktöryelini hesapla. sayıyı giriniz"))
        Faktöryel = 1
        For i = 1 To N Step 1
            Faktöryel = Faktöryel * i
        Next i
        MsgBox(N & " sayısının faktöryeli " & Faktöryel & " dir.")
End Sub
```

3.4. Girilen bir sayıya kadar olan doğal sayıların toplamını hesaplama (Toplam=ΣN=0+1+..+N)

Programın Akış Diyagramı;

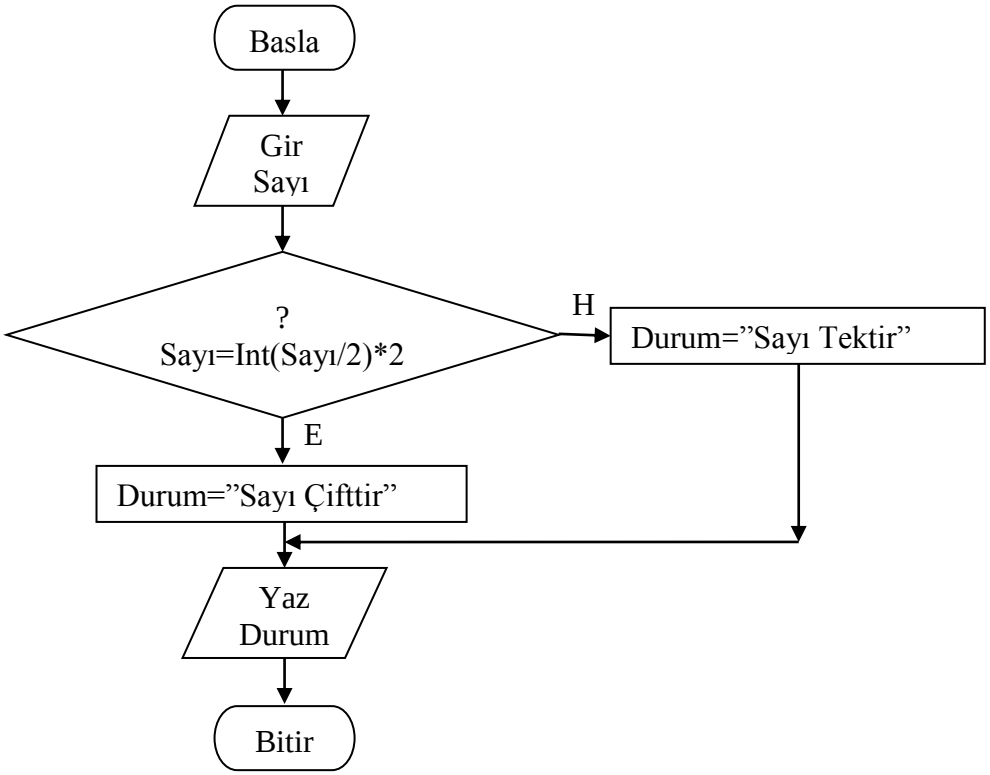


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        N = Val(InputBox("N sayıyı giriniz"))
        Toplam = 0
        For i = 1 To N Step 1
            Toplam = Toplam + i
        Next i
        MsgBox(N & " sayısına kadarki doğal sayıların toplamı " &
Toplam & " dir.")
End Sub
```

3.5. Girilen bir sayının tek/çift olup olmadığının belirlenmesi

Programın Akış Diyagramı;

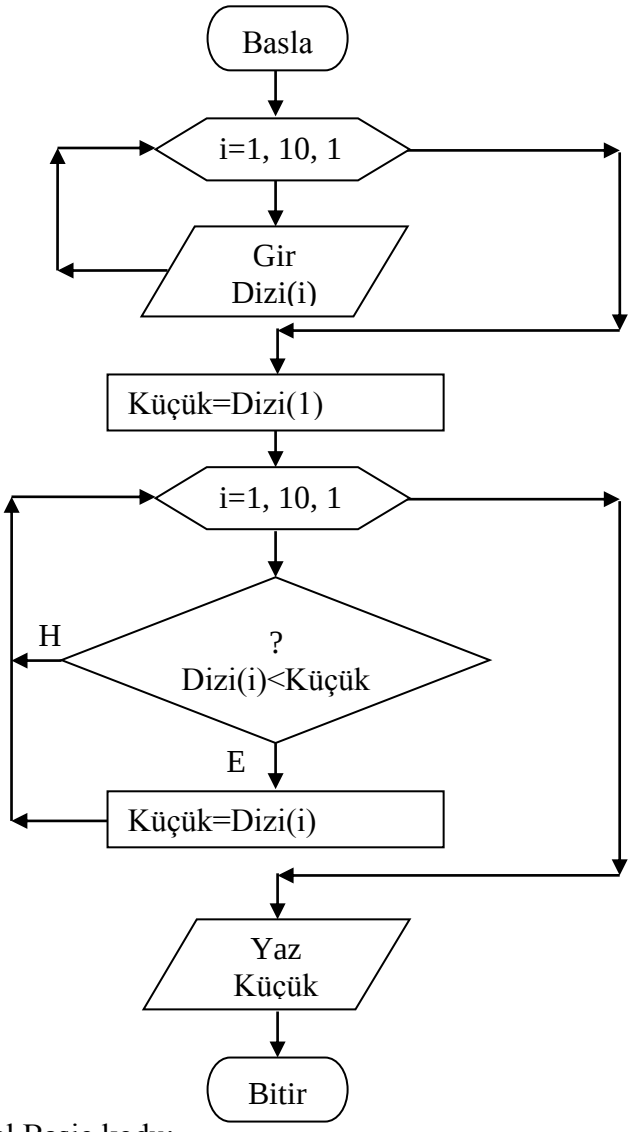


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayı = Val(InputBox("Sayıyı giriniz"))
        If Sayı = Int(Sayı / 2) * 2 Then
            Durum = "Sayı Çifttir"
        Else
            Durum = "Sayı tektir"
        End If
        MsgBox(Durum)
End Sub
```

3.6. Bir dizi içerisindeki sayıların en küçüğünün/büyüğünün belirlenmesi (Dizi(10)=[20,18,2,0,-2,1,1,0,30,-1])

Programın Akış Diyagramı;

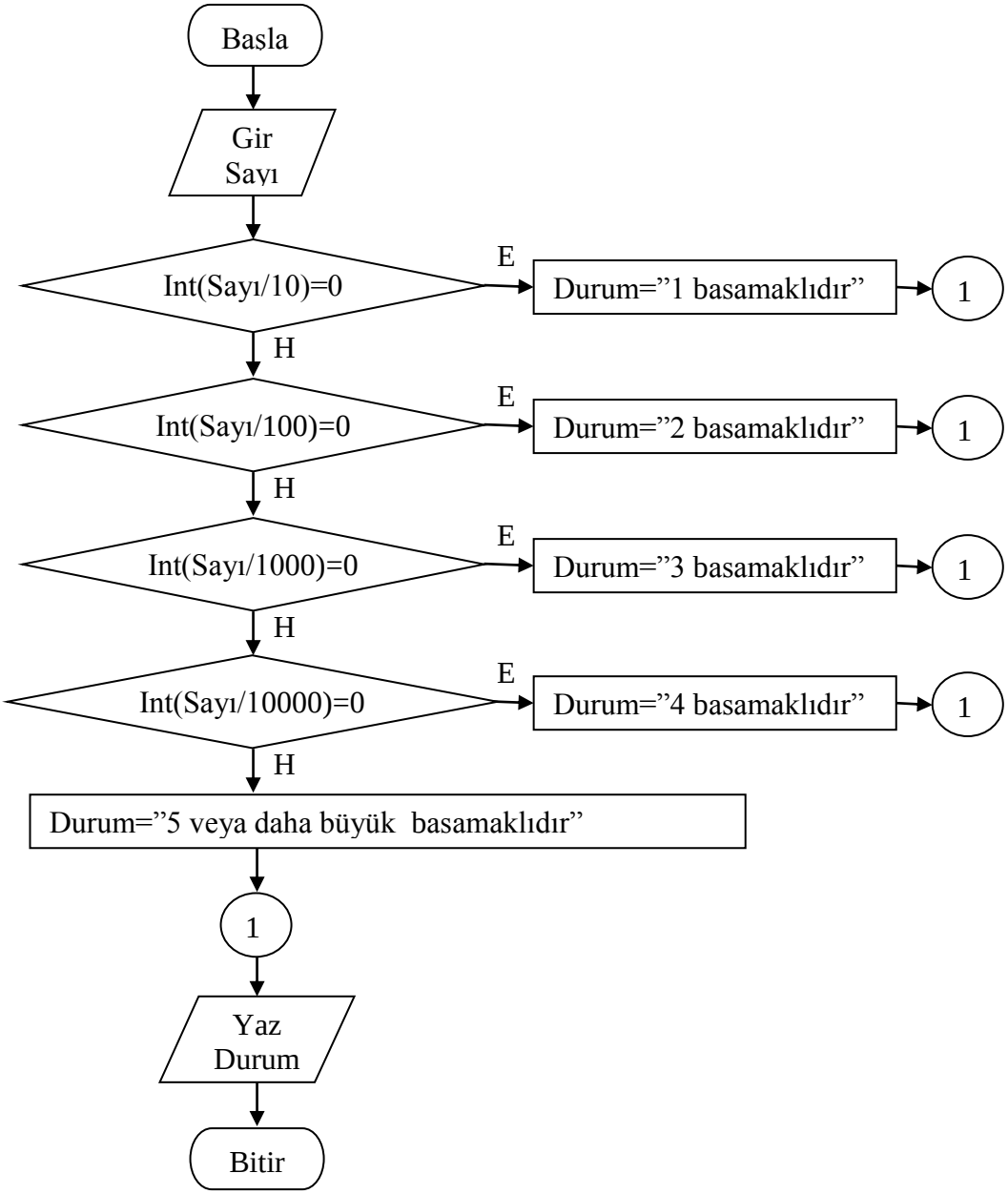


Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Dim Dizi(10)
        For i = 1 To 10
            Dizi(i) = Val(InputBox("Dizinin elemanını giriniz "))
        Next i
        Küçük = Dizi(1)
        For j = 1 To 10
            If Dizi(j) < Küçük Then
                Küçük = Dizi(j)
            End If
        Next j
        MsgBox(Küçük)
End Sub
```

3.7. Girilen herhangi bir sayının kaç basamaklı bir sayı olduğunun belirlenmesi

Programın Akış Diyagramı;



Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Sayı = Val(InputBox("Sayıyı giriniz"))
        If Int(Sayı / 10) = 0 Then
            Durum = "1 basamaklıdır"
        ElseIf Int(Sayı / 100) = 0 Then
            Durum = "2 basamaklıdır"
        ElseIf Int(Sayı / 1000) = 0 Then
            Durum = "3 basamaklıdır"
        ElseIf Int(Sayı / 10000) = 0 Then
            Durum = "4 basamaklıdır"
        Else
            Durum = "5 veya daha büyük basamaklıdır"
        End If
        MsgBox(Sayı & " Sayısı " & Durum)
End Sub
```

3.8. Satır sayısı (N) ve Sütun sayısı (M) girilen iki boyutlu (NxM elemanlı) bir matrisin bütün elemanlarına 1 değerinin atanması.

Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
FileOpen(10, "C:\Users\User\Desktop\deneme.txt", OpenMode.Output)
        Dim Matris(,)
        N = Val(InputBox("Matrisin satır sayısını giriniz"))
        M = Val(InputBox("Matrisin sütun sayısını giriniz"))
        ReDim Matris(N, M)
        For i = 1 To N
            For j = 1 To M
                Matris(i, j) = 1
            Next j
        Next i
        For i = 1 To N
            For j = 1 To M
                Print(10, Matris(i, j))
            Next j
            PrintLine(10)
        Next i
        FileClose(10)
        MsgBox("İşlem Bitti")
End Sub
```

3.9. Eleman sayısı girilen NxN elemanlı bir kare matrisinin ana köşegeni ve altındaki elemanlarına 0 değerinin diğerlerine 1 değerinin atanması.

Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
FileOpen(10, "C:\Users\User\Desktop\deneme.txt", OpenMode.Output)
        Dim Matris(,)
        N = Val(InputBox("Matrisin satır sayısını giriniz"))
        ReDim Matris(N, N)
        'Matrisin bütün elemanlarına 1 değerinin atanması
        For i = 1 To N
            For j = 1 To N
                Matris(i, j) = 1
            Next j
        Next i
        'Matrisin köşegenleri ve altındaki eleman. 0 değ. atanması
        For i = 1 To N
            For j = 1 To N
                If i = j Then
                    Matris(i, j) = 0
                    Exit For
                Else
                    Matris(i, j) = 0
                End If
            Next j
        Next i
        'Elde edilen matrisin yazdırılması
        For i = 1 To N
            For j = 1 To N
                Print(10, Matris(i, j))
            Next j
            PrintLine(10)
        Next i
        FileClose(10)
        MsgBox("İşlem Bitti")
End Sub
```

3.10. Kullanıcıdan ara, dönemsonu sınav sonuçlarını alıp başarı notunu (başarı=ara*0.4+dönemsonu*0.6) hesaplayarak, ara, dönemsonu ve başarı notlarının yazdırılması.

Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
FileOpen(10, "C:\Users\User\Desktop\deneme.txt", OpenMode.Output)
        Ara = Val(InputBox("Arasınav notunu giriniz"))
        Dönemsonu = Val(InputBox("Dönemsonu notunu giriniz"))
        Başarı = Ara * 0.4 + Dönemsonu * 0.6
        Print(10, Ara, Dönemsonu, Başarı)
        FileClose(10)
        MsgBox("İşlem Bitti")
End Sub
```

3.11. Fibanacci serisi olarak bilinen sayılar; 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, … vb. şeklinde devam eder. Her sayı kendisinden önce gelen iki sayının toplamıdır. Bu durumda genel olarak n'inci Fibonacci sayısı F(n) şu şekilde ifade edilir.

$$F_n = F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

Bu açıklamalara göre ekrandan girilen n değerine göre F(n)'in değerinin belirlenmesi

Programın Visual Basic kodu;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
FileOpen(10, "C:\Users\User\Desktop\deneme.txt", OpenMode.Output)
        Dim Fibonacci()
        n = Val(InputBox("n değerini giriniz"))
        ReDim Fibonacci(0 To n)
        For i = 0 To n
            If i = 0 Then
                Fibonacci(0) = 0
            ElseIf i = 1 Then
                Fibonacci(1) = 1
            Else
                Fibonacci(i) = Fibonacci(i - 1) + Fibonacci(i - 2)
            End If
        Next i
Print(10, n & ". Fibonacci sayısı " & Fibonacci(n) & " dir.")
        FileClose(10)
        MsgBox("İşlem Bitti")
End Sub
```

4. VİSUAL BASİC NESNELERİ

4.1. Projeler

Projenin özellikleri Project / Properties menüsünden görüntülenir. *General/ StartupObject* kısmında *Sub Main* veya ilk çalışacak form adı olmalıdır. *Sub Main* yordamı modülde olmalıdır, Tools/*add procedure* modüle eklenir.

App Nesnesi: Proje hakkında bilgi edinmede kullanılır.

App.EXEName : Çalışan exe dosyasının adını verir. [*Msgbox App.EXEName*]

App.Title: Task Manager deki listede yazılı olan adı değiştirir. [*App.Title="Notlar"*]

App.Path: Çalışma anında geçerli olan yol bu değişkende saklanır.

App.PrevInstace: Program çalışırken bir kez daha çalıştırılmak istenirse, bunu engellemek için kullanılabilir.

Örnek;

```
Private Sub Form_Load()
If App.PrevInstace Then
Msgbox "Bu program şu anda çalışıyor"
End
End If
End Sub
```

4.2. Formlar

4.2.1. Formların Özellikleri (Properties)

Formların özellikleri Properties penceresinde değiştirileceği gibi, kod yazarak da değiştirilebilir. Bu bilgiler *formunadı.özellik* isimli değişkenlerde saklanır.

Caption: Formun başlık bilgisi.

MinButton: True veya False değerini alır. Simge durumunda küçültme özelliğini denetler.

MaxButton: True veya False değerini alır. Ekranı kapla özelliğini denetler.

ControlBox: True veya False değerini alır. Formun (pencere) sol üst köşesinde bulunan menüyü denetler.

BorderStyle: Pencerenin boyutlarının ayarlanabilirliğini denetler.

Width, Heigh: Twip olarak formun genişliğini ve yüksekliğini belirlerler.

Left, Top: Formun masaüstündeki yerini belirlerler.

FontName: Formun font tipini belirler. [*FontName= "Arial"*]

FontBold, FontItalic, FontUnderline, FontStrikeThru: True veya False değerini alırlar. sırasıyla koyu, italic, altıçizili, üstüçizili özelliklerini belirlerler.

Picture: Formun zeminine resim dosyası ekler, resmin boyutları formla birlikte değişir. [*Form1.Picture=LoadPicture("dosyaadı")*]

BackColor: Formun zemin rengini belirler. Rengi belirlerken RGB(kırmızı, yeşil, mavi) fonksiyonu kullanılabilir. [0 – 255] arası değer alabilirler.

WindowState: Formun başlangıç durumunu belirler.

Icon: Formun ikonunu belirler.

ScaleMode: Form dahilinde kullanılabilecek ölçü birimini belirler.

ScaleLeft, ScaleTop: Mevcut koordinat sisteminin orijinini yeniden tanımlar.

MousePointer: Fare işaretinin şekliyle ilgili bilgiyi içerir.

Enabled: True veya False değerini alır. Formun kullanılabilirliğini belirler.

DrawWidth: Formun üzerine Line() yada Circle() ile çizim yapılırken kullanılan çizginin kalınlığını belirler.

4.2.1. Formlara Uygulanan Olaylar (Events)

Olay meydana geldiğinde hazır yordamları kullanırız, olaylar kod olarak da girilebilirler.

Load ( ): Form belleğe yüklenmesi olayı.

Unload (*Cancel* ***):*** Formun bellekten silinmesi olayı. *Cancel* parametresi default "-1" değerine sahiptir. *Cancel = 1* olursa *Unload* işlemi gerçekleşir.

Click ( ) : Formun üzerine mouse ile tek tıklama olayı.

DblClick ( ): Formun üzerine mouse ile çitf tıklama olayı.

Activate ( ): Formun aktif hale gelmesi olayı.

Deactivate ( ): Formun pasif hale gelmesi olayı.

Resize ( ): Formun boyutlarında değişiklik yapılması olayı.

KeyPress (*KeyAscii as Integer* ***):*** Klavyeden (Shift, Ctrl ve Alt haricinde) bir tuşa basma olayı, *KeyAscii* parametresi basılan tuşun ascıı kodunu yakalar.

KeyDown (*KeyCode As Integer, Shift As Integer* ***):*** Klavyeden bir tuşa basıldığında, basılı durumda olduğunda çalışır, *KeyDown* olayının *KeyPress* olayına göre önceliği vardır. *KeyCode* parametresi basılan tuşun ascıı kodunu yakalar, *Shift* parametresi ise Shift, Ctrl ve Alt tuşlarına basılı olup olmadığını araştırır. (tabloya bakınız.)

Shift' in Degeri	Basılı Durumda Olan Tuşlar
0	Shift, Ctrl ve Alt tuşlarından hiçbirine basılı değil
1	Shift tuşuna basılı durumda
2	Ctrl tuşuna basılı durumda
3	Shift ve Ctrl tuşlarına basılı durumda
4	Alt tuşuna basılı durumda
5	Shift ve Alt tuşlarına basılı durumda
6	Ctrl ve Alt tuşlarına basılı durumda
7	Shift, Ctrl ve Alt tuşlarına basılı durumda

KeyUp (*KeyCode As Integer, Shift As Integer****):*** *KeyDown* olayı gibidir, tuşun serbest bırakıldığı sırada çalışır.

MouseDown (anında *Button as Integer, Shift As Integer, X As Single, Y As Single* ***):*** Bu olay çalışma formun üzerinde farenin herhangi bir tuşuna basıldığında meydana gelir. Farenin soldaki tuşuna basmışsa *Button = 1*alır. *Shift* , sağdaki tuşuna basmışsa *Button = 2*, her ikisine birden basılmışsa *Button = 3* değerini parametresinin kullanımı yukardaki gibidir. işaretinin *X* ve *Y* ' parametresine bu olay meydana geldiğinde fare form üzerindeki konumu hakkında twip cinsinden bilgi aktarılır.

MouseUp (*Button as Integer, Shift As Integer, X As Single, Y As Single****):*** Bu olay farenin basılı olan tuşundan el kalkınca meydana gelir. Parametrelerin kullanımı ise *MouseDown* olayı ile aynıdır.

MouseMove(*Button as Integer, Shift As Integer, X As Single, Y As Single****):*** Fare işaretinin form üzerindeki yeri değiştiği zaman meydana gelir. Parametrelerin kullanımı ise *MouseDown* olayı ile aynıdır.

Paint(): Bu olay formun üzerine başka bir form geldiğinde ve formun boyutlarında değişiklik yapıldığında meydana gelir. Altta kalan forma daha önce Print, Line vs. gibi bir deyimle bilgi yazımı yapılmışsa bazı bilgiler kaybolabilir, bunu engellemek için kullanılabilir.

4.2.3. Formlara Uygulanan Methodlar

Cls: Form üzerine print vs. ile yazılanları silmek için kullanılır.

Line (*Başlama X, Başlama Y****)-(****Bitiş X, Bitiş Y****), ÇizgiRengi, BF:*** Belirtilen noktalar arasında çizgi çizer, başlama yeri belirtilmez ise o anki *CurrenX* ve *CurrentY* değerleri baz alınır. *ÇizgiRengi RGB(kırmızı, yeşil, mavi)* şeklindedir. "*B*" yalnız kullanılırsa belirtilen noktalar köşeler olmak üzere bir dikdörtgen çizilir, "*F"* de kullanılırsa bu dikdörtgenin içi boyanır.

Circle (*Merkez X, Merkez Y****),*** *Yarıçap****,*** *ÇizgiRengi****,*** *Başlama****,*** *Bitiş****:*** Belirtilen merkez, yarıçap ve renkte çember çizer. Başlama default olarak "0" ve Bitiş de "6.28" yani 2p değerine sashiptir.

Pset (*X, Y* ***),*** *NoktaRengi****:*** Belirtilen koordinatlara bir nokta yerleştirir.

DrawWidth = *sayı* ***:*** Çizgi yada nokta için çizim kalınlığını ayarlar.

Move(*SolÜstKöşe* ***X,*** *SolÜstKöşe* ***Y):*** Formu hareket ettirmek için kullanılır, bunun için formun sol üst köşesinin ekrandaki değeri twip cinsinden girilir.

Refresh: Formun gerektiğinde tazelenmesi ve güncellemesi için kullanılır.

TextWidh ( ) *ve* ***TextHeight ( ):*** Parantez içine text yada değişken yazılır. Belirli bir textin formun üzerinde twip cinsinden ne kadar yer kaplayacağını bildiren terimlerdir.

4.2.4. Mdi Form ve Child formlar

Aynı anda birden fazla formu açık tutmak için MDI (Multiple Document Interface) formlar kullanılır. *Project* menüsünden *add MDI form* komutu ile eklenir. Bir projede bir tane MDI form olabilir, ancak birçok alt form *(child)* içerebilir. Herhangi bir form, *MDIChild* özelliği *true* değeri yapılarak *child* durumuna getirilebilir. *Child* formlar MDI formun penceresinin içersinde yer alırlar. MDI özellikli formlardan pencerenin sağ-üst köşesindeki *Minimize, Maximize, Close* komutları kaldırılamaz. MDI formlara menu, *PictureBox* ve *ToolBox* dışında nesne dahil edilemiyor. Yinede MDI formun üst kısmına *PictureBox* nesnesi ekledikten sonra –başka yere eklenmiyor- bu kısma diğer nesneler eklenebilir. Gerekirse *ToolBox* yapılabilir.

Çalışma anında projeye yeni Child Formlar Eklemek: Projede *Form1* adında *Child* form olduğunu varsayalım.

< *Dim Belge As New Form1* > satırıyla *Form1* yapısına ve özelliklerine sahip *Belge* isminde yeni bir *Child form* oluşturulur.

Child Formların MDI Formlar içindeki yerleşimi *Arrange* methodu ile olur:

MDI formun adı. Arrange <n> (n = 1, 2, 3)

1, ise yatay dizilirler

2, ise dikey dizilirler

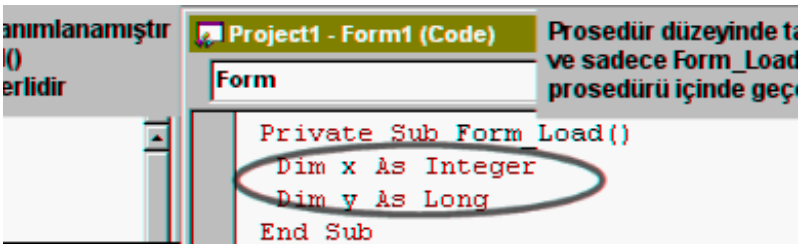
3, ise üst üste dizilirler.

Projeye hazır formlar ve şablonlar eklenebilir, bunun için *Project / Add* Form dan sonra *Optins Dialog, Log In Dialog* vs. seçilebilir.

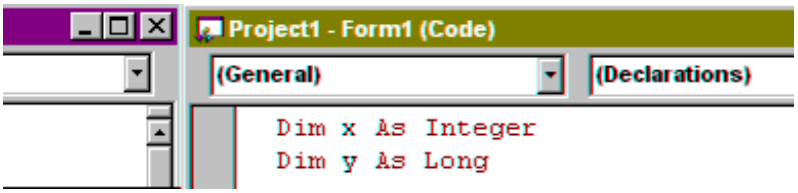
Formlara özel kontroller eklenebilir. *OCX* uzantılı dosyalar özel kontrol dosyalarıdır. Menüden *Project / Components* seçilerek özel kontroller için *OCX* dosyası seçilir. Formlara özel kontroller eklenirse *OCX* uzantılı dosyaların da projeye eklenmesi gerekir.

5. DEĞİŞKENLERİN GEÇERLİLİK BÖLGESİ

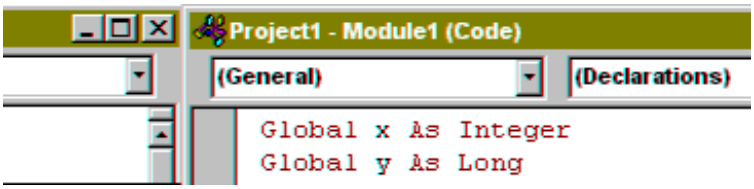
Aksi Belirtilmedikçe VB de tanımlanan değişkenler kullanıldığı prosedür içinde geçerlidir. Diğer bir ifadeyle değişkenler bulundukları prosedürler içinde yerel (lokal) değişkenlerdir. Aşağıdaki örnekleri inceleyiniz



Eğer değişkenin bir prosedür içinde değil de form düzeyinde geçerli olması istenirse formun General, Declaration kısmına yerleştirmek gerekir.



Son tanımlana şekliyle x ve y değişkenleri Form düzeyinde tanımlanmıştır. Ancak halen x, y değişkenleri Private (yani özel) olup Form içerisinde geçerlidir. Bu değişkenlerin Projenin tamamında tanımlamak için Modül düzeyinde tanımlamak gerekir . Bunun için Modül penceresi açılır ve General, Declaration kısmına Global [değişken] [türü} biçiminde tanımlama yapılır.



6. STATİK VE DİNAMİK DEĞİŞKENLER

Aksi belirtilmedikçe tüm yerel değişkenler dinamiktir, yani olay prosedürünün her çağırılışında, bir önceki çağırılış sonundaki değerlerini saklamazlar, bunun yerine ilk atanan değerleri ile çalışmaya başlarlar. Değişkenin değerini saklar hale getirmek için, Static sözcüğü kullanılır. Örneğin, Static x As Integer biçiminde bir tanımlamayla x değişkeni prosedürün her çağılışında, bir önceki değerini saklar hale gelecektir. Bunu daha iyi anlayabilmek için aşağıdaki iki örneği yazarak aradaki farkı anlamaya çalışınız.

Örnek:

Form üzerinde bir komut buttonu ve bir text kutusu yerleştiriniz.Ve komut buttonu üzerine çift clickleyerek aşagıdaki kodu giriniz ve Run/Start ile çalıştırınız

```
Private Sub Command1_Click()
Dim w As Integer
w = w + 10
TextBox1.Text = w
End Sub
```

Command1 buttonuna her basışınızda Text1 kutusunda 10 yazdığını göreceksiniz çünkü Dim ile tanımladığımız w değişkeni dinamiktir. Şimdi yukarıdaki kodda bulunan Dim sözcüğünü Static olarak değiştirin ve programı tekrar çalıştırın değişikliği gördünüz mü?

7. TİP DÖNÜŞÜMLERİ

VB'de kullandığımız sayısal değerleri diğer veri tiplerine dönüştürebiliriz. Bu işlemi yapan fonksiyonlara tip dönüşüm fonksiyonları denir. Tip dönüşüm fonksiyonları aşağıda verilmiştir.

CBool(ifade) Matemetiksel ifadeyi Boolean türüne dönüştürür.
CByte(ifade) Matemetiksel ifadeyi Byte türüne dönüştürür.
CCur(ifade) Matemetiksel ifade Currency türüne dönüştürür.
CDate(ifade) Matemetiksel ifade Date türüne dönüştürür.
CDbl(ifade) Matemetiksel ifade Double türüne dönüştürür.
CDec(ifade) Matemetiksel ifadeDecimal sayıya dönüştürür.
CInt(ifade) Matemetiksel ifade tam sayıya dönüştürür.
CLng(ifade) Matemetiksel ifade Long türüne dönüştürür.
CSng(ifade) Matemetiksel ifade Single türüne dönüştürür.
CVar(ifade) Matemetiksel ifade Variant türüne dönüştürür.
CStr(ifade) Matemetiksel ifade String türüne dönüştürür.

Aşağıdaki örnekleri inceleyiniz;

Örnek1:

A=12 , B=12 , C=6 , D=0

```
deger = CBool(A < B) 'deger=False
deger = CBool(A > C) 'deger=True
deger = CBool(A = C) 'deger=False
deger = CBool(D) 'deger=False
deger = CBool(Not D) 'deger=True
```

Örnek2 :

A=10 , B=5 , C=0

deger = CByte(A < C) 'deger=0
deger = CByte(A > C) 'deger=1
deger = CByte(A = B) 'deger=0

Örnek3 :

A=1 , B=2 , C=36000 , D=36001

deger = CDate(A) 'deger=12/31/1899
deger = CDate(B) 'deger=1/1/1900
deger = CDate(C) 'deger=7/24/98
deger = CDate(D) 'deger=7/26/98

8. GEÇMİŞ YILLARDA YAPILAN SINAV SORULARI

MDZ110 Bilgisayar Programlama

(Arasınav – 07/04/2014)

S.1. İki metin kutusu(**TextBox1** ve **TextBox2**) ile ekrandan girilecek olan MIN ve MAX değerlerine göre; X'in MIN'in aldığı değerden MAX'ın aldığı değere kadar 0.2 artışlarla alacağı değerlere karşılık gelen Y değerini aşağıda verilen eşitliğe göre hesaplayıp, sadece Y değerini negatif yapan X ve Y değerlerini C sürücüsündeki "sonuclar.txt" dosyasına yazdıran programın akış diyagramını çiziniz ve Visual Basic diliyle kodlayınız(60 P).

$$Y = \frac{2X^3 - 200X + 2}{e^{x+1}}$$

S.2. Belirtilen veri dosyasına göre aşağıda verilen altprogram (program) çalıştırıldığında değişkenlerin hafızada ve ekranda aldığı değerleri sırayla takip ederek belirtiniz (40 P).

```
Sub Program ( )
FileOpen(2, "Degerler.txt" OpenMode.Input)
FileOpen(3, "Ekran.txt" OpenMode.Output)
Dim AA(0 To 2, 0 To 2), BB(0 To 2)
For i = 0 To 2
  Input(2, BB(i))
  Input(2, Deger)
  For j = 0 To 2
    Input(2, AA(i,j))
  Next j
Next i
For k = 0 To 2 Step 1
  TopBB = TopBB + BB(k)
  TopAA1 = TopAA1 + AA(k, 1)
Next k
PrintLine(3, TopBB)
PrintLine(3, TopAA1)
PrintLine(3, "İşlem bitti!")
 End Sub
```

"**Degerler.dat**"

```
------- Top Of  File ------
-2
8
6
5
2
0
-9
12
6
0
-2
1
11
-6
1
------ End Of  File ---------
```

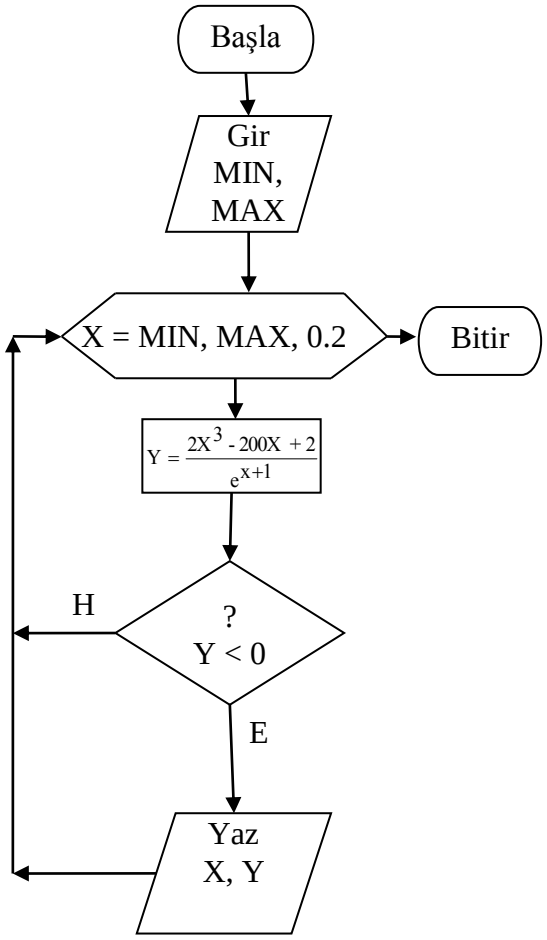
Sınav süresi **60** dakikadır. Başarılar dilerim.

Doç.Dr. Ahmet DAĞ

`Math.Exp( ), FileOpen( ), OpenMode.Input/Output`

MDZ110 Bilgisayar Programlama(Arasınav Cevaplar – 07/04/2014)

C.1.




```
Private Sub Button1_Click()
FileOpen(10, "C:\sonuclar.txt",
OpenMode.Output)
MIN = Val(TextBox1.Text)
MAX = Val(TextBox2.Text)
For X = MIN To MAX Step 0.2
Y = (2 * X ^ 3 - 200 * X + 2) /
Math.Exp(X + 1)
If Y < 0 Then
Print(10, X, Y)
End If
Next X
FileClose(10)
End Sub
```

C.2.

HAFIZA

i	BB	Deger	j	AA	k	TopBB	TopAA1
~~0~~	(0)=-2	~~8~~	~~0~~	(0,0)=6	~~0~~	~~-2~~	~~5~~
~~1~~	(1)=0	~~-9~~	~~1~~	(0,1)=5	~~1~~	~~-2~~	~~11~~
~~2~~	(2)=-2	1	~~2~~	(0,2)=2	~~2~~	-4	5
3			~~3~~	(1,0)=12	3		
			~~0~~	(1,1)=6			
			~~1~~	(1,2)=0			
			~~2~~	(2,0)=11			
			~~3~~	(2,1)=-6			
			~~0~~	(2,2)=1			
			~~1~~				
			~~2~~				
			3				

EKRAN

```
-4
5
İşlem bitti!
```

MDZ110 Bilgisayar Programlama 26/5/2014 (Arasınav Mazeret)

S.1. Ekrandan girilen Nesne Tipi(NT; Prizma için 1, Silindir için 2, Koni için 3 ve Küre için 4) ve bu nesneye ait yine ekrandan girilen bilgilere göre nesnenin hacmini hesaplayıp ekrana yazdıran bu şekilde hacim hesaplamasını NT için 0 girilinceye kadar devam ettirip 0 girildiğinde işlemi sonlandıran bir programın akış diyagramını çiziniz (30 P).

$\text{Prizma}(a,b,c) = a \cdot b \cdot c$, $\text{Silindir}(r,h) = \mu \cdot r^2 \cdot h$, $\text{Koni}(r,h) = \frac{1}{3} \cdot \mu \cdot r^2 \cdot h$, $\text{Küre}(r) = \frac{4}{3} \cdot \mu \cdot r^3$

S.2. **InputBox** ile ekrandan girilen 4 haneli bir tamsayının(N), binler(BIN), yuzler(YUZ), onlar(ON) ve birler(BIR) basmağındaki rakamları bulup bunların toplamını **TextBox1.Text**'e yazdıran bir programı Visual Basic diliyle yazınız. (Algoritmayı kurabilmenize yardımcı olmak amacıyla açıklama; örneğin N sayımız 5617 olsun; bu sayının BIN rakamı 5617/1000 işlemi sonucunun tamsayı kısmı olan 5, YUZ rakamı (5617-5000)/100 işlemi sonucunun tamsayı kısmı olan 6, ON rakamı (5617-5000-600)/10 işlemi sonucunun tamsayı kısmı olan 1 ve BIR rakamı ise (5617-5000-600-10) işleminin sonucu olan 7 olur. Bir ifadenin tamsayı değeri ise **Int**(İfade) hazır fonksiyonu ile elde edilebilir) (35 P)

S.3. Belirtilen veri dosyasına göre aşağıda verilen program (altprogram) çalıştırıldığında değişkenlerin hafızada ve ekranda aldığı değerleri sırayla takip ederek belirtiniz(35 P).

```
Sub Program ( )
FileOpen(2,"Degerler.dat", OpemMode.Input)
Dim A(0 To 7), B(0 To 7)
For i = 0 To 7 Step 1
  Input(2, A(i))
  Input(2, B(i))
Next i
For j = 1 To 7
   Top = A(j-1) + B(j-1)
   Durum="Toplam(" & j-1 & ")=" & Top
   MsgBox(Durum)
   Next j
End Sub
```

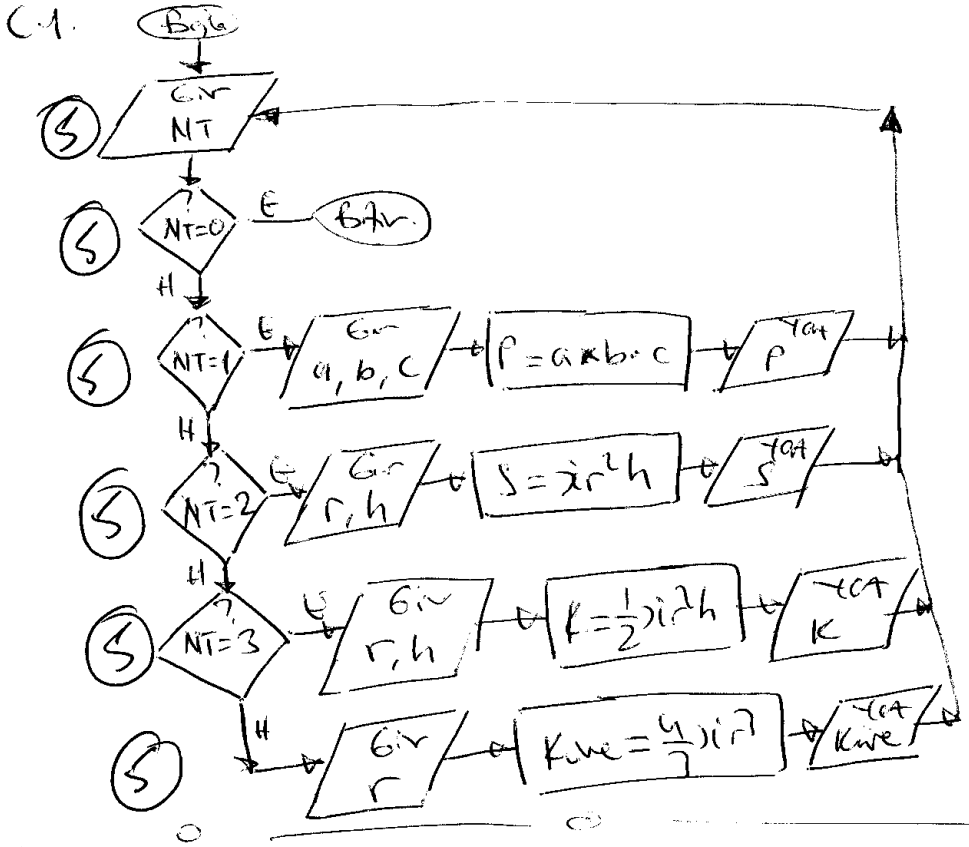
"**Degerler.dat**"

```
----- Top Of  File -----
-2
18
6
9
2
0
-9
21
6
0
-2
1
1
1
----- End Of  File ------
```

Sınav süresi **60** dakikadır. Başarılar Dilerim.

Doç.Dr. Ahmet DAĞ

C-1.



C-2.

```
Sub ...
N = Val(InputBox(" "))
BIN = Int(N/1000)
YUZ = Int((N - BIN*1000)/100)
ON = Int((N - BIN*1000 - YUZ*100)/10)
BIR = N - BIN*1000 - YUZ*100 - ON*10
Text1.Text = BIN + YUZ + ON + BIR
End Sub
```

C-3

Hafıza

i	A	B	j	Top
1	(1) = -2	(1) = 15	2	16
2	(2) = 6	(2) = 9	3	15
3	(3) = 2	(3) = 0	4	2
4	(4) = -9	(4) = 21	5	13
5	(5) = 6	(5) = 0	6	6
6	(6) = -2	(6) = 1	7	-1
7	(7) = 1	(7) = 1	8	2
8	(5)	(5)	9	(5)
(5)				

Ekran

Toplam(1) = 16
Toplam(2) = 15
Toplam(3) = 2
Toplam(4) = 13
Toplam(5) = 6
Toplam(6) = -1
Toplam(7) = 2

(10)

MDZ110 Bilgisayar Programlama (Final Sınavı – 10/6/2014)

S.1. Aşağıda verilen y(t) fonksiyonunun, t'nin 3, ile 15 aralığında 3'er artışlarla aldığı değerlere göre y(t) fonkisyonunun değerlerini hesaplayıp hem t'nin hem de y nin değerini "ekran.txt"dosyasına yazdıran programın akış diyagramını çiziniz ve Visual Basic diliyle kodlayınız (40P).

$$y(t)=\begin{cases} e^{t+0.1} & , \quad t \le 5 \\ \sqrt{t^2}-10 & , \quad 5 < t \le 10 \\ 2 & , \quad t > 10 \end{cases}$$

S.2. **InputBox** ile ekrandan sırayla girilen 10 sayıdan (Sayı) en küçük olan sayıyı bulup bu sayıyı form üzerindeki metin kutusuna(**TextBox1.Text**) yazdıran bir bilgisayar programını Visual Basic diliyle yazınız (20 P).

S.3. Aşağıda kodu verilen Command1 kliklendiğinde (çalıştırıldığında) ve **InputBox** ile sırayla 5, 4, -2 değerleri girildiğinde değişkenlerin hafızada ve ekranda aldığı değerleri sırayla takip ederek belirtiniz(40P).

```
Sub Command1_Click()
  FileOpen(1, "sonuclar.txt", OpenMode.Output)
  10 a = Val(InputBox("deger giriniz"))
  If a <= 0 Then
       PrintLine(1, "İşlem bitti")
       End
  Else If Int(a / 2) * 2 = a Then
       b = Toplam(a + 1)
       PrintLine(1, b)
  Else
       c = Carpım(a - 2)
       PrintLine(1, c)
  End IF
  GoTo 10
End Sub
       Function Toplam(N)
            Toplam = 0
            For j = 1 To N
            Toplam = Toplam + j
            Next j
       End Function
       Function Carpım(M)
            Carpım = 1
            For i = 1 To M
            Carpım = Carpım * i
            Next j
       End Function
```


`Math.Exp( ), Math.Sqrt( ), FileOpen( ), OpenMode.Input/Output`