

Algoritma ve Bilgisayar Programlama

Mapbasic Programlama Dili Kullanarak Haritacılıkta Programlama

Öğr. Gör. EMRE İNCE
KAMAN MESLEK YÜKSEKOKULU

Doküman, Harita ve Kadastro Programı Öğrencilerinin programlama dili kullanarak program tasarımını kavramaları, harita yapımı amaçlı algoritma kurma veya program oluşturmalarını öğrenmeleri, Coğrafi Bilgi Sistemleri II dersinde öğrenilecek bilgileri kullanarak internet ortamında harita paylaşımı ve haritacılık amaçlı cep telefonu uygulamaları tasarımı yapabilmeleri için temel algoritma yapısını öğrenmeleri hedeflenmiştir. Ders, öğrencilerin programlama kavramalarını daha kolay anlamaları için, Mapinfo Coğrafi Bilgi Sistemi yazılımını geliştirmekte kullanılan Mapbasic Programlama Dilini ve hazırlanmış kütüphanelerini kullanılarak anlatılacaktır. Dokümanda belirlenen hedef doğrultusunda, Mapbasic kullanımı, harita özelliklerinin Mapbasic sayesinde geliştirildiğini, haritacılıkta kullanılan mesleki hesaplamaların Mapbasic yazılımıyla oluşturulması konuları işlenmektedir.

Öğr. Gör. Emre İNCE
Kaman Meslek Yüksekokulu
Harita ve Kadastro Programı

İÇİNDEKİLER

MapBasic Programının Temel Kullanımı.....	1
Değişkenler ve Veri Tipleri.....	6
Değişkene Değer ataması.....	8
Mapbasic Dilinde Kullanılan Hazır Fonksiyonlar	10
Sayının Üst Kuvvetinin Alınması	10
Sayının Karekökünün Bulunması	11
İki Nokta Arasındaki Yatay Mesafe (Kartezyen Mesafe)	12
Ondalık hane sayısı belirleme (Şekil 12).....	12
Açı birimleri ve Trigonometrik fonksiyonlar	12
Mapbasic dilinde kullanılan trigonometrik fonksiyonlar:	15
İç içe fonksiyonların kullanımı	17
Semt açısının Hesaplanması	17
Karşılaştırma Yapıları.....	17
İf – Then Karşılaştırma Yapısı.....	18
Karşılaştırma Operatörleri:.....	19
Mantıksal Operatörler:	19
ElseIf Yapısı.....	20
Not operatörünün kullanımı.....	21
İki nokta arasındaki Semt Açısının Hesaplanması	21
Procedur (Yordam) ve Function (Fonksiyon) Kavramı	22
Yordam Tanımlanması ve Çalışma Prensipleri	23
Program Yönetiminde Ana Yordamın Kullanılması	24
Karekök Yordam Örneği.....	26
İçine Değer Alan (Parametre alan) Yordam Tanımlanması.....	26
Açı Değerini Derece – Dakika – Saniye Açılış Birimine Dönüştürüp Ekranaya Yazan Yordam	27

Fonksiyon Tanımlanması ve Fonksiyon Kullanımı Örneği.....	28
Sayının Kuvvetini Alan Fonksiyon Örneği.....	29
Açı Değerini Derece – Dakika – Saniye Açılış Birimine Dönüştürüp Ekranaya Yazan Fonksiyon.....	30
İki Nokta Arasındaki Yatay Mesafenin Hesabını Yapan Fonksiyon	31
Form Tasarımı ve Kullanıcı Ara Yüzüyle Programlama.....	32
MapBasic Geliştirme Ortamı MapBasic IDE	32
MapBasic IDE Programının Bilgisayara Kurulması İşlem Adımları	33
Formda Event (Olay) Kontrolü	37
Girilen Değerlerin Yazdırılması	39
Buton Kullanımı	39
Kullanıcı tarafından girilen değerlerin kontrolü ve Yordam Kullanımı	42
Program Kod İçinde Hata Yakalama İşlemleri.....	45
Girilen Rasyonel Sayı Halindeki Derece Açılış Birimini Derece – Dakika – Saniye Haline Dönüştürülmesi ve Yordam Kullanımı	47
Kullanıcıdan Alınan Reel Sayı Veri Tipindeki Alan Değerinin Alt Birimlere Ayrılması	50
Kullanıcıdan Alınan Veri kullanılarak Hesaplama	52
Kullanıcıdan Alınan Veriler ile Hesaplama (Hem Yordam Hem de Fonksiyon Kullanımı).....	55
RadioGroup Kullanımı ve Yan Nokta Hesabı	59
Döngü.....	67
For – Next Döngü Yapısı.....	67
Do – Loop Döngü Yapısı.....	68
Buton (Düğme) Araçları	68
Buton Tipleri Örneği.....	69
Haritada Tıklanan Yerin Koordinatlarını Ekranaya Yazan Buton Programı	72
Çizgi veya Çoklu Doğru Grafik Objelinin Uzunluk Bilgisi.....	73

CartesianDistance() Fonksiyonunun Kullanımı.....	74
Distance(x1,y1,x2,y2, birim) Fonksiyonu:	77
CartesianObjectLen() fonksiyonu:	77
Çokgen grafik objesinin Alan Değerinin Hesaplanması.....	81
Haritada Seçilen Nokta Grafik Objesinin Koordinatlarını Yazdıran Program	83
Grafik Objenin Harita Düzlemine Eklenmesi ve Objenin Kayıt Altına Alınması.....	85
Grafik Objenin Ek Bilgi Olmadan Haritaya Eklenmesi	86
Kullanıcıdan Alınan Koordinat Değerleriyle Nokta Objesi Ekleme.....	86
Create Point Komutlarıyla Haritaya Nokta Ekleme	86
Kullanıcının Çizdiği Çizgi Objesinin Tabakaya Eklenmesi	91
Create Line() Komutlarıyla Çizgi Grafik Objesinin Eklenmesi.....	91
Grafik Objenin Ek Bilgileriyle Beraber Harita Penceresine Eklenmesi.....	93
Structure Query Language (SQL) Komutlarıyla Haritaya Nokta Ekleme	93
Kullanıcının Nokta Ekleme veya Noktanın Koordinat Bilgilerinin Güncellemesini Yapan program.....	98
Structure Query Language (SQL) Komutlarıyla Çizgi Objesini Eklenmesi.....	107
Structure Query Language (SQL) Komutlarıyla Haritaya Çoklu Doğru Ekleme.....	110
Kullanıcının Çizdiği Çokgen (Alan) Grafik Objesinin Tabakaya Eklenmesi.....	111
Birden Fazla Aracı Tek Bir ButtonPad içinde toplama	117
Menü Eklenmesi	119
Menü Eklenmesi İşlemi	119
Elektronik Takeometre ile Yapılan Ölçüm verilerinin Eklenmesi	120
Geriden Kestirme Hesabı.....	126
Yatay Mesafe Değeri Kullanılarak Geriden Kestirme Hesabı	126
Metin Dosyalarının Okunması.....	130
Mapinfo Yazılımında Oluşturulmuş Haritanın Tablolarından Veri Çekilmesi	136
SQL Parametreleri.....	136

Tablodan Veri Seçimi İçin SQL cümlesi Kurulması (Select Parametresi)	136
Tablo Verilerinin Sahalara Göre Gruplandırılması	145
Tablodaki Verilerin Sahalara Göre Sıralanması	146
SQL Parametrelerinin Mapbasic Kodları İçinde Kullanımı	147
Haritadaki Grafik Objeler ile Çalışılırken Bilinmesi Gereken Fonksiyonlar	154
Set Coordsys.....	154
Frontwindow()=	155
LayerInfo()=	155
Windowinfo()=.....	158
Kullanımı:.....	158
Objectinfo()=.....	161
Kullanımı=.....	161
SearchInfo()=	162
Kullanımı=.....	162
Ek -1Semt açısının Hesaplama adımları:.....	164
Ek – 2 Yan Nokta Hesabı	166

MapBasic Programının Temel Kullanımı

MapBasic, Basic Programlama dilini temel alan ve MapInfo Coğrafi Bilgi Sistemi yazılımını geliştirmek için içerisine kütüphaneler eklenmiş bir derleyicidir. Derleyici yazılan programlama dilini satır satır okur, kod satırları içinde hata olup olmadığını kontrol eder ve hata bulursa kullanıcıyı uyarır, hata olmayan program kodunun çalıştırılabilir (Executable) dosyasını üretir. Çalıştırılabilir dosyalar genelde Exe uzantılıdır. Mapbasic ile derlenen programın çalıştırılabilir dosya uzantısı Mbx'dir.

Mapbasic, Mapinfo yazılımının geliştirilmesi için oluşturulduğundan yazılan programın çalışması tek başına yeterli değildir. Mapinfo programının da Mapbasic programının kurulu olduğu bilgisayara kurulması gereklidir. Mapbasic içinde yazılan program çalıştırıldığında Mapinfo programı da otomatik açılacaktır.

MapBasic içindeki kullanılacak programlama dilinin öğrenirken ilk olarak programın kullanım mantığını anlamak gerekli. Aşağıda hazırlanmış olan Uygulama 1 örneği programın kod yazımı mantığı, programın derlenmesi ve çalıştırılabilir dosyasının oluşturulması işlemlerini anlamak için hazırlanmış bir örnektir.

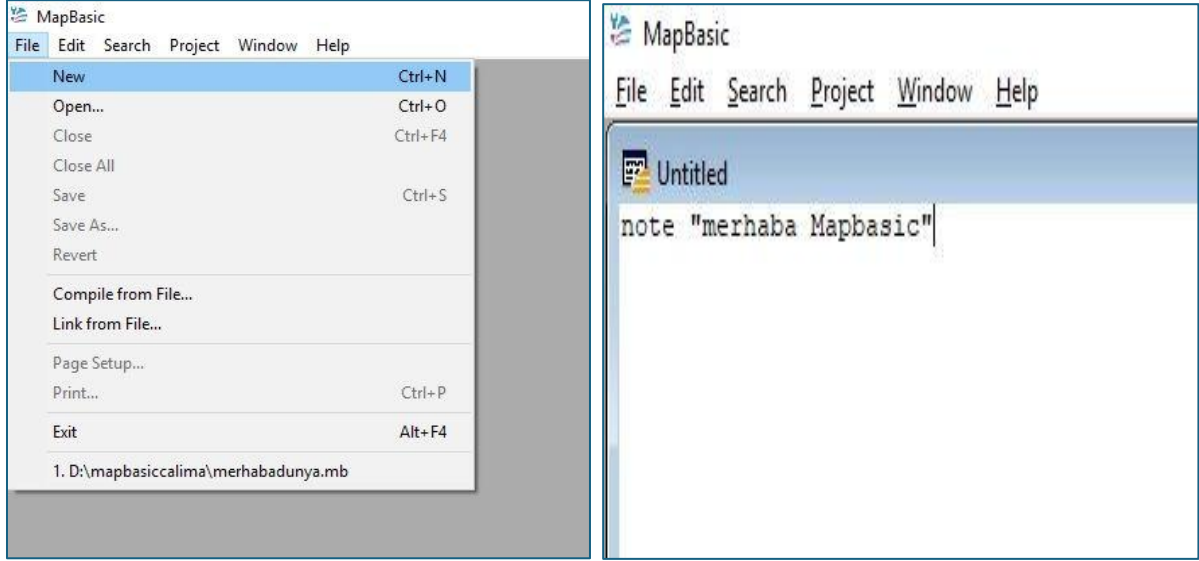
Uygulama 1

İlk uygulamada, Mapbasic penceresi içinde kod yazma ortamı ve yazılan kodun çalıştırılması anlatılacaktır. Mapbasic, Basic ve Visual Basic dillerini temel alan kodlama diline sahiptir. İlk yapılacak çalışmada, Mapinfo ekranında bir mesaj ekranı çıkartılması ve istenilen bir iletinin mesaj ekranına yazılması olacak. Bunun için **Note** komutu kullanılacak

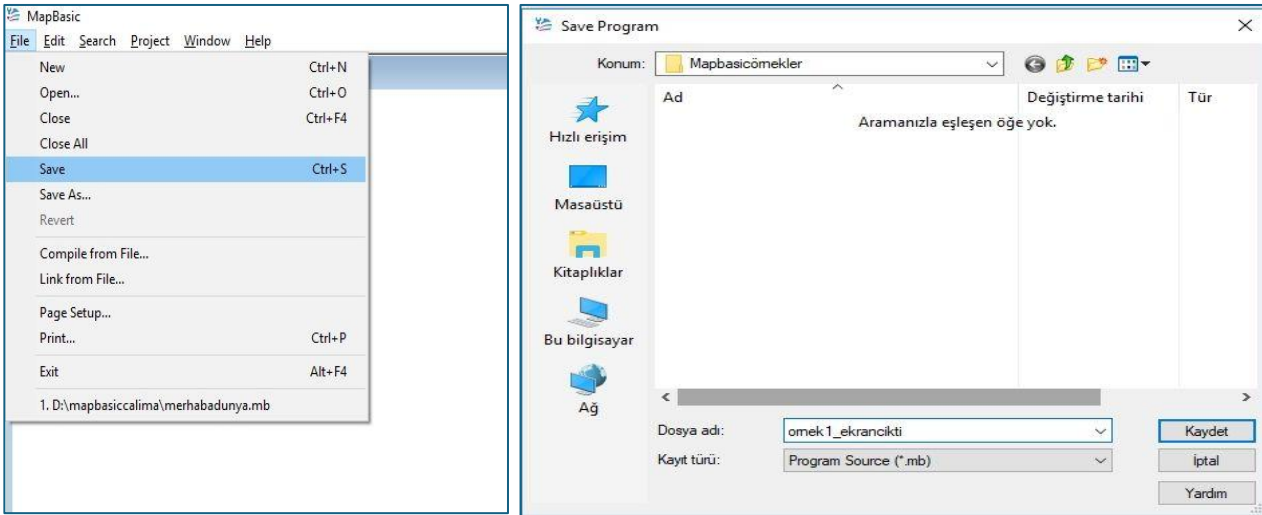
Bilgisayarda başlat menüsü altındaki Programlar arasından Mapbasic çalıştırıldığında Şekil 1 soldaki resimde yer alan Mapbasic penceresi açılacak. Açılan pencerede New (yeni) menüsü seçildiğinde Şekil 1 sağ pencerede yer alan Untitled (isimsiz) adlı dökümanda ilk satıra Şekil 1 sağ pencerede olduğu gibi

Note "merhaba Mapbasic"

Yazılıp File (dosya) menüsünden dosyamızı kaydetmek için Save (kaydet) tuşuna basılır (Şekil 2).

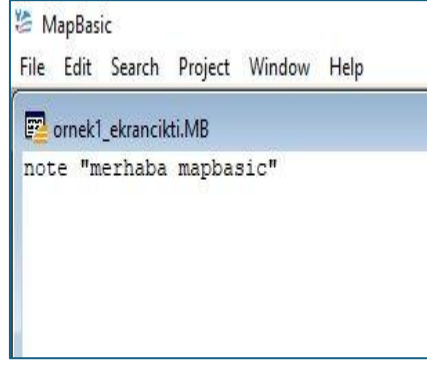


Şekil 1



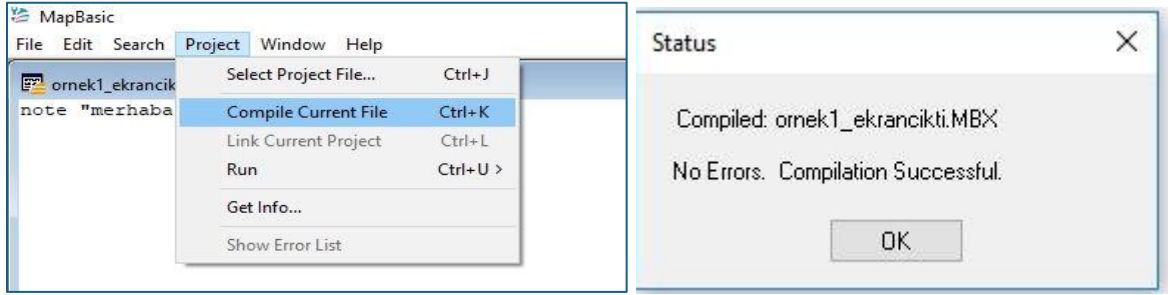
Şekil 2

Dosyayı kaydetmek için açılan Kaydet penceresinde **Dosya adı** kısmına **ornek1_ekrancikti** yazıp kaydet tuşuna basılması gereklidir. Kaydet işleminden sonra kod yazdığımız ekran başlığı dosyamızın ismine dönüşecek ve kaydet işlemi sırasında bilgisayarda belirttiğimiz dizin içinde MB uzantılı dosyamız oluşacaktır (Şekil 3).



Şekil 3

Kayıt işleminden sonra yazdığımız kodun hatalarının bulunması ve hata yoksa kodun çalıştırılabilir dosyaya dönüştürülmesi için **derlenmesi** gerekir. Kodun derlenmesi için Project (proje) menüsü alt menülerinden **Compile Current File** (Aktif Dosyayı Derle) seçilir (Şekil 4sol resim). Eğer hata yoksa Şekil 4 sağ resimde olduğu gibi **No Errors. Compilation Successful** (Hata yok. Derleme Başarılı) mesaj penceresi ekrana gelir.



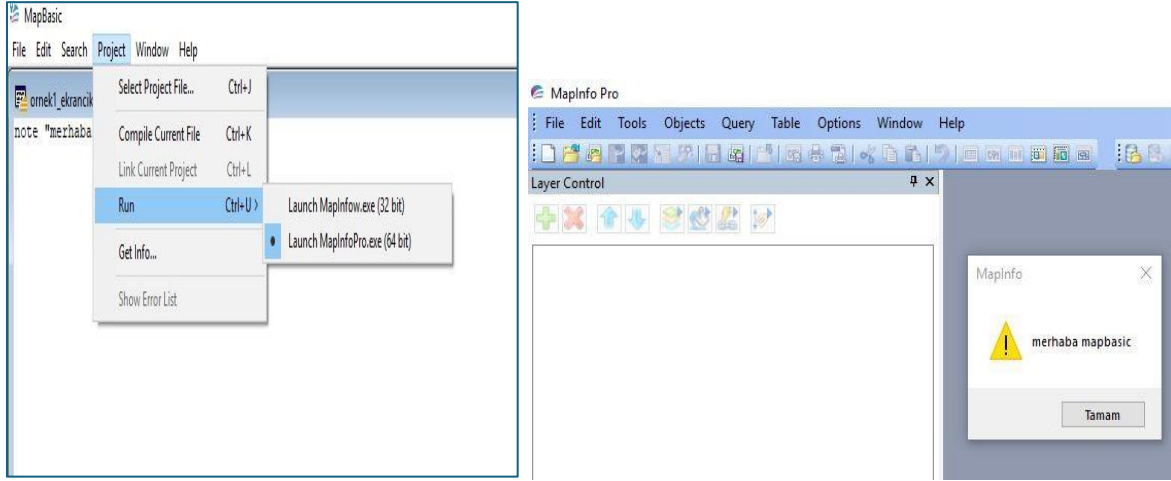
Şekil 4

Tamam tuşuna bastığımızda kod dosyamızla aynı isimli ve kod dosyamızın kayıtlı olduğu dizinde **MBX** uzantılı çalıştırılabilir dosya oluşur (Şekil 5). Bu dosyaya çift tıklandığında ya da Project menüsünden Run (Koştur / Çalıştır) seçildiğinde programımız çalışacaktır.

Mapbasicörnekler			
Ad	Değiştirme tarihi	Tür	Boyut
ornek1_ekrancikti.MB	27.01.2018 19:07	MapBasic Progra...	1 KB
ornek1_ekrancikti.MBX	27.01.2018 19:08	MapBasic Applica...	1 KB

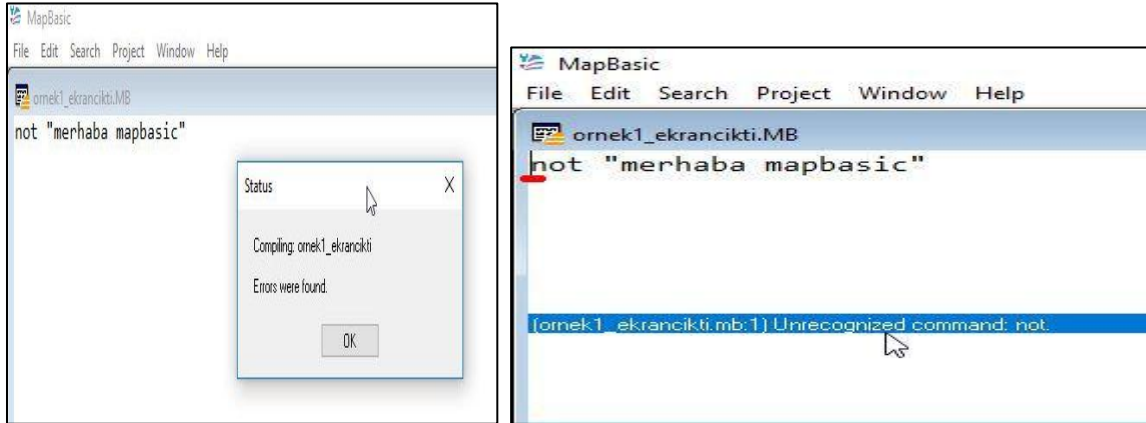
Şekil 5

Şekil 6 sağ resimde programın çalışmış hali gözükmetedir. Program çalışırken eğer kurulu olan Mapinfo yazılımı kapalıysa programı açacaktır ve ekrana bir mesaj kutusuyla ekrana istenilen yazı çıkacaktır.



Şekil 6

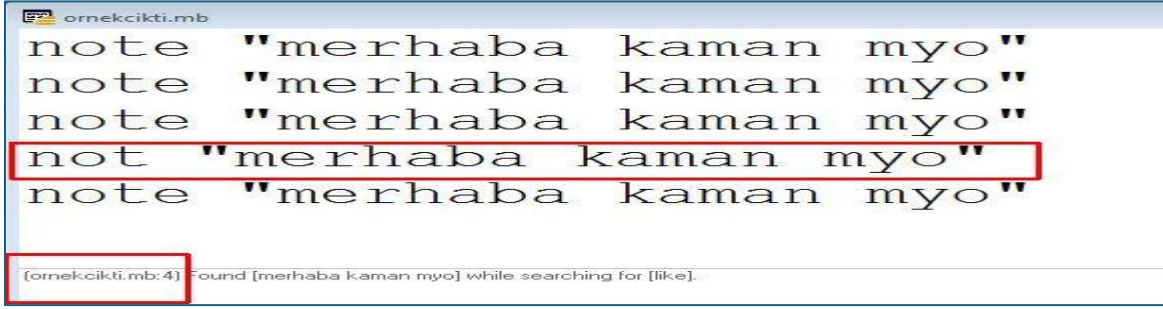
Yazılan program kodu derlendikten sonra oluşan hatalar derleyici tarafından bulunup listelenir. Şekil 7 sol pencere derleme sonrası **Error were found (Hatalar bulundu)** mesajı ekranda belirmiş. **Tamam** tuşuna basıldığında ekranın altında hataların hangi satırda olduğunu belirten bir mesaj beliriyor (Şekil 7sağ resim). Hata koduna baktığımızda mb:1 ifadesinde ki 1 birinci satırı gösteriyor. Hata mesajı olan **“Unrecognized command (tanımlanmamış komut): not”** not komutunun daha önce tanımlı olmadığını ifade etmiş. Eğer Şekil 7 sağ resimde olduğu gibi hata mesajı üzerinde çift tıklarsanız hatanın olduğu satıra fare imlenicini konumlandıracaktır. Binlerce satırlık bir kod yazıldığında hatanın bulunmasını kolaylaştırır.



Şekil 7

Uygulama 2

MapBasic derleyicisinde hatalı yazılan kodun olduğu satırın gösterilmesi. Şekil 8 hatalı kod yazımına bir örnektir. Aynı satır birden fazla kez alt alta yazılmış ve dördüncü satırda note komutu yanlış yazılmıştır. Program derlendiğinde derleyici kaçınıcı satırda hata olduğunu belirtmektedir.



```

ornekcikti.mb
note "merhaba kaman myo"
note "merhaba kaman myo"
note "merhaba kaman myo"
not "merhaba kaman myo"
note "merhaba kaman myo"

[ornekcikti.mb:4] Found [merhaba kaman myo] while searching for [like].

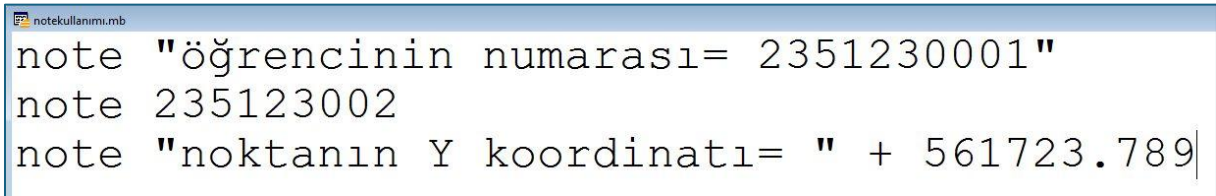
```

Şekil 8

Uygulama 3

Note komutunda "" (iki çift tırnak) aralığı metin ifadelerin gösteriminde kullanılmaktadır. Metin ifade olarak anlatılmak istenen hem alfabetik hem de sayısal değerlerin olduğu ifadelerdir. Eğer ifade sadece sayısal bir değer olarak kullanılmak isteniyorsa "" kullanılmasına gerek yoktur.

Şekil 9 "" aralığında ifadelerin kullanımına bir örnek yapılmıştır. İlk satırda "" aralığında hem alfabetik hem de sayısal değerler kullanılmıştır. İkinci satırda "" kullanılmamış sadece sayısal değer kullanılmıştır. Sadece sayısal değerler kullanılacaksa "" kullanılmasına gerek yoktur. Üçüncü kod satırında "" aralığında metin ifade kullanılmış ve sonra sayısal ifade kullanılmış. Üçüncü satırdaki kodun çalışabilmesi için "" bitimine + parametresi kullanılmıştır. + parametresi, metin ifade ile sayısal ifadeyi toplama değil birleştirme amacıyla kullanılmıştır.

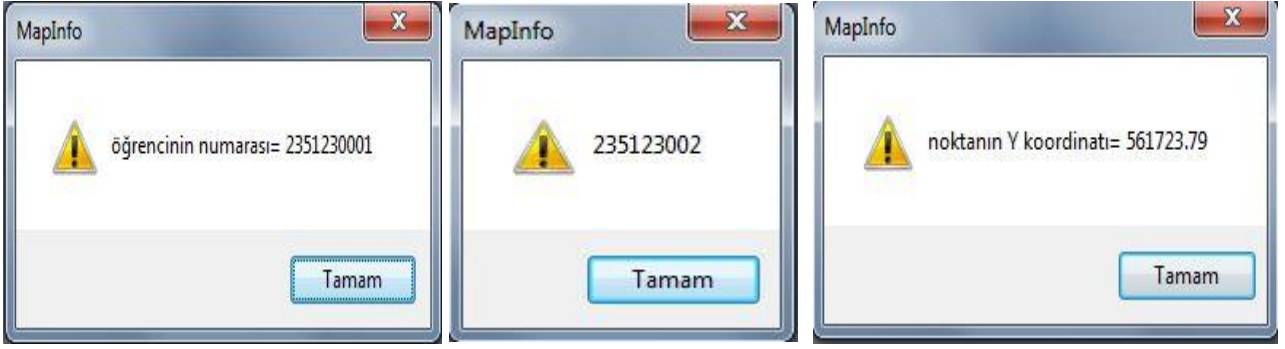


```

notekullanimi.mb
note "öğrencinin numarası= 2351230001"
note 235123002
note "noktanın Y koordinatı= " + 561723.789

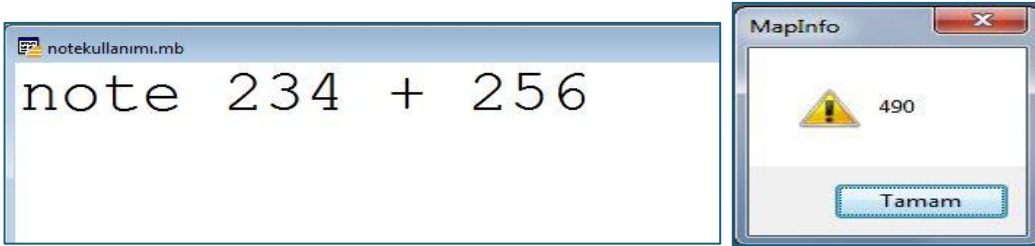
```

Şekil 9



Şekil 10

Note ifadesinde " " kullanılmadan sadece sayısal değerler ve bu değerler arasında + ifadesi varsa, program + ifadesini toplama olarak algılar.



Şekil 11

Değişkenler ve Veri Tipleri

Değişkenler yazılan programın kullanıcıdan istediği değerleri tutmada (saklamada) veya yazılacak program içinde kullanılacak değerleri (verileri) tutmada kullanılır. Matematikte kullanılan bilinmeyenler veya kullandığımız hesap makinesindeki hafıza tuşları gibi düşünülebilir. Değişkenler kullanıcı tanımlıdır ve içerisine hangi tip değeri alacağı önceden belirlenir. Kullanıcı tanımlı ifadesinden anlaşılması gereken değişkene bir isim verilmesi ve değişkenin içine alacağı verinin tipinin kullanıcı tarafından belirlenmesidir.

Değişkenler Tablo 1 değişken tipinde belirtilen veri tiplerinde tanımlama yapılırken Basic dilleriyle aynı **Dim** operatörü kullanılır. ifade 1 değişken tanımına örnektir. Değişken adını kullanıcı belirler ve değişken adı belirlemenin belirli kuralları vardır. Değişkenin tutacağı verinin türüne göre değişken tipi belirlenmesi gerekir.

ifade 1

dim degisken_adi as degisken_tipi

Mapbasic programlama dilinde kullanıcının tanımlayacağı değişkenler Tablo 1 içerisinde gösterilen değişken veri tiplerine göre değer alabilirler. Değişken tipinden kastedilen değişkenin içinde tutacağı verinin sayısal, sözel, tarihsel, mantıksal veya grafik obje tipinde olduklarını belirtir:

Tablo 1

Değişken Tipi	Değişken Tanımı
SmallInt	-32767 ile +32767 sayıları arasında kalan tam sayıları tutmak için kullanılır.
Integer	-2 milyar ile +2 milyar arasında kalan tam sayıları tutmak için kullanılır.
Float	Rasyonel sayıları tutmak için kullanılır.
String	32767 tane karakter alabilen metin verileri tutmak için kullanılır.
String *n	Sabit n uzunluğunda metin ifadeleri tutmak için kullanılır. n değeri 32767'ye kadar artabilir.
Logical	True (doğru) veya False (yanlış) mantıksal değerleri tutmak için kullanılır.
Date	Tarih verisini tutmak için kullanılır.
Object	Çizgi, nokta, alan gibi grafik objeleri tutmak için kullanılır.
Alias	Tablolardaki sahaların isimlendirilmesinde kullanılır.
Pen	Çizgi stil ayarları için kullanılır.

Örneğin bir binanın köşe noktalarına ad verilecekse String veri tipi kullanılır. Çünkü noktanın adı 1 olabileceği gibi B1 de olabilir. Noktanın X ve Y koordinatları değişkende tutulacaksa koordinatları tutacak değişkenin veri tipi float veri tipinde olacaktır. Çünkü koordinatlar reel sayı tipindedir. Eğer çizilen obje bir değişkende tutulmak isteniyorsa değişkenin veri tipi object olmalıdır.

Dim x As Integer (1 .sattır)

dim y,z as integer (2 .sattır)

dim soz as string, tarih as date (3 .sattır)

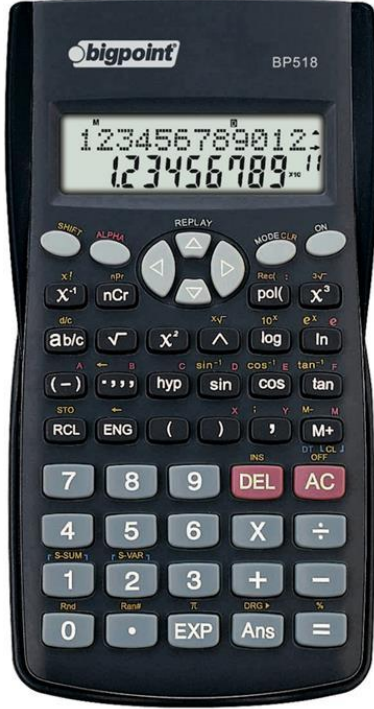
Üstteki üç satırda değişken tanımlamasına örnekler verilmiştir. Her satırda yapılan tanımlama doğrudur. Aralarına virgül konularak birden fazla aynı tipte farklı değişken tanımlanabilir. 3. Satırda metin ifade tutacak **soz** değişkeni tanımlandıktan sonra virgül (,) konularak farklı veri tipinde yeni bir değişken tanımlanabilir.

Değişken tanımlanırken değişken isimlerinde dikkat edilmesi gerekenler:

- ❖ Değişken adları azami 31 karakterden oluşabilir,
- ❖ Değişken adları içerisinde boşluk olamaz (örneğin: Dim ad soyad as String →tanımlaması yanlıştır),

- ❖ Değişken adları alt çizgi (_) veya Tilda (~) karakterleriyle başlayabilir (örneğin: Dim _x1 as Integer → şeklinde tanımlama yapılabilir),
- ❖ Değişken adları Mapbasic programlama dilinde tanımlı bazı kelimelerden oluşamaz (örneğin Then, Select, Open, Close veya Count kelimeleri değişken adı olamaz.)

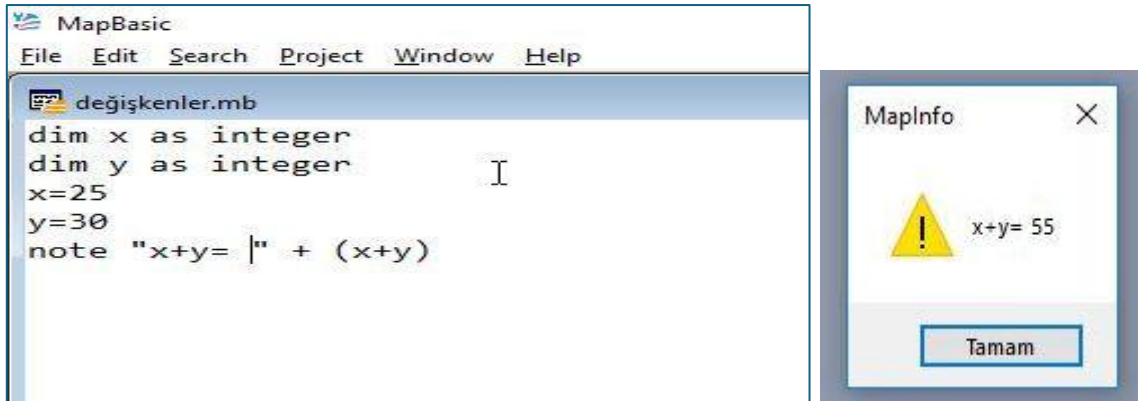
Değişkene Değer ataması



Hesap makinesi kullanırken A, B, C, D, E, F, X, Y ve M adlı değişkenler mevcuttur. Bu değişkenler reel sayı veri tipinde tanımlanmıştır. Hesap makinesinde tanımlı değişkenlere değer ataması yapılırken sabit işlemler vardır. Örneğin A değişkenine değer ataması yapılırken shift →sto→A işlemi yapılır.

Mapbasic'de kod yazımında tanımlanmış değişkenlere değer atamak için = operatörü kullanılır. Şekil 12 sol resimde Mapbasic editöründe görüldüğü üzere x ve y adında iki adet tam sayı tipinde değişkene değer ataması gözükmektedir. Kodun en

sonunda iki tam sayının toplamı ekrana yansıtılmıştır. Kod derlenip çalıştırıldığında Şekil 12 sağ resimde görüldüğü gibi oluşacaktır.



Şekil 12

Mapbasic derleyicisinde yazılan programlama dili python, c, c++, java, c#,... gibi programlama dilleri için kullanılan derleyicilere göre daha esnek bir derleyicidir. Örneğin

kullanıcının tanımladığı sayısal veri tutacak bir değişkene metin veri aktarılmasında derleyici hata vermelidir. Şekil 13’de görülen kod bahsedilen örneğin uygundur. `parsel_no` isminde tanımlanan değişken tamsayı veri (integer) tutacağı belirlenmiştir.

İkinci satırda `parsel_no` değişkenine 24.2 reel sayı değeri atanmıştır. Tamsayı bir değişkene ondalıklı sayı değeri atanmamalıdır. Fakat Mapbasic 24.2 değerinin sadece tam sayı kısmını alacaktır (Şekil 24 sol resim). Benzer şekilde 4. Satırda `parsel_no` değişkenine 15a metin ifadesi aktarılmıştır. Mapbasic 15a metin ifadesinin ilk değerleri sayısal olduğu için sadece 15 değerini algılar. 6. Satırda `parsel_no` değişkenine metin ifade aktarılıyor. Mapbasic bu hatayı algılamıyor. Hata program çalıştığı zaman ortaya çıkıyor. Hata program kodu yazılırken fark edilmelidir. Yazılıp çalışır dosya haline getirilen program kullanıcı tarafından kullanılacak. Kullanıcı çıkan hatayı anlayamaz ve sorunu çözemez. Sorunu programcı fark etmelidir. Kullanılan derleyicinin program kodu içindeki hatayı anlayamaması olumsuz bir durumdur. Hatta 10000 satırlık bir program kod bütünü içinde hata bulmak daha da zor olacaktır.



Derleyicinin hata bulamayacağını düşünerek program kodu içinde tanımlanan değişkenlerin tutacağı verilere göre değişken tipi belirlenmeli ve değişkene değer ataması yapılırken kontrol mekanizmaları kurulmalıdır.

```
MapBasic - [veritipi.mb]
File Edit Search Project Window Help
dim parsel_no as integer
parsel_no=24.2
note parsel_no
parsel_no="15a"
note parsel_no
parsel_no="abc"
note parsel_no
```

Şekil 13

Şekil 14 sağ resim oluşan hatayı göstermektedir. Hata Mapinfo programında çıkmaktadır.



Şekil 14

Aynı hesap makinesinde olduğu gibi aynı değişken birden fazla veri ataması yapılabilir. Yukarıdaki örnekte `parsel_no` değişkenine peşi sıra birden fazla değer ataması yapılmıştır. Değişkenin en son tuttuğu veri yapılan son veri atamasındaki veridir.

Mapbasic Dilinde Kullanılan Hazır Fonksiyonlar

Fonksiyon içerisine veri alan veri geriye sonuç veri döndüren matematiksel işlem bütünüdür. ifade 2 içinde f adlı fonksiyon içine x adlı değişkenden veri alır, x veri f fonksiyonu için işleme sokulur, geriye y isminde değişkene değer atamasıyla sonucu geri döndürür.

ifade 2

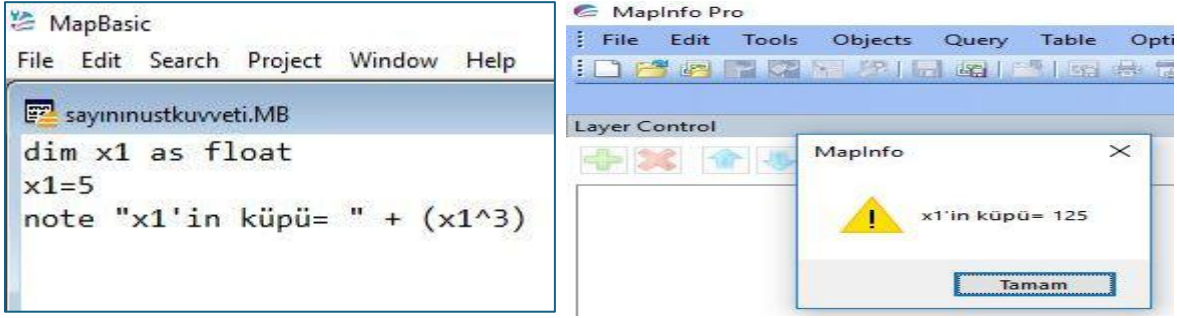
$$y = f(x)$$

Bazı fonksiyonlar geriye değer döndürmez bu tip fonksiyonlar yordam olarak isimlendirilir.

Mapbasic derleyicisi içinde kullanılan programlama dilinde, metin ve sayısal verilerle işlemlerin yapılabilmesi için hazır fonksiyonlar ve yordamlar bulunmaktadır. Hesap makinesin de sayısal işlemlerin yapılması için hazır fonksiyonlar bulunmaktaydı. Örneğin üst alma, karekök, sinüs, cosinüs, tanjant trigonometrik fonksiyonları, sayının tersinin alınması gibi fonksiyonlar hesap makinesinde hazır olarak bulunmaktadır. Mapinfo içinde kullanılan programlama dilin de hazır fonksiyonlar bulunmaktadır. Fonksiyonlar hem sayısal veriler üzerinde işlem yapmak için oluşturulmuş hem de metin ifadeler üzerinde işlem yapmak için oluşturulmuştur.

Sayının Üst Kuvvetinin Alınması

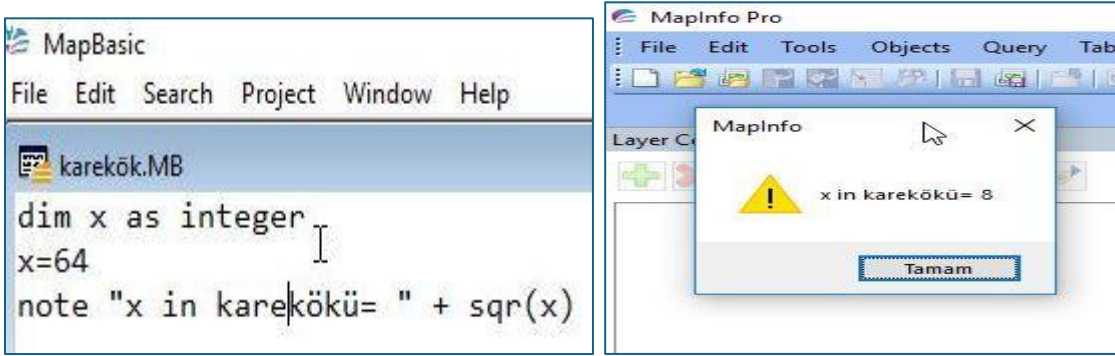
Mapbasic derleyicisinde yazılan kod içinde üst ifadesi kullanılacaksa $^$ ifadesi kullanılır. Şekil 15 üstlü ifadenin bir örneğidir.



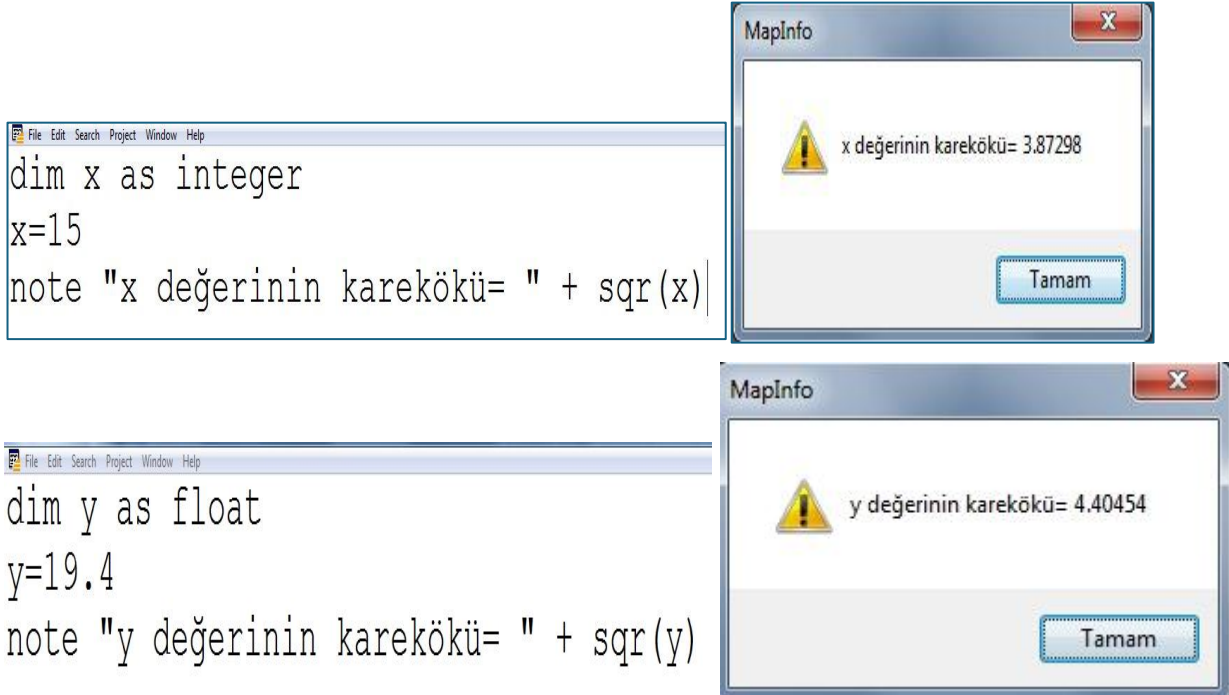
Şekil 15

Sayının Karekökünün Bulunması

SQR() fonksiyonu, içine aldığı değerin karekökünü float (rasyonel sayı) olarak geri döndürür.

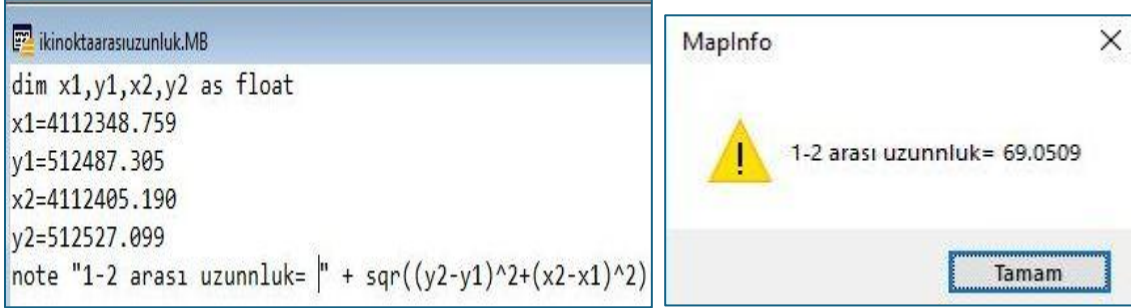


Şekil 16



İki Nokta Arasındaki Yatay Mesafe (Kartezyen Mesafe)

Üst ve karekök fonksiyonları öğrenildiğine göre haritacılıkta temel ödev olarak bilinen iki nokta arasındaki mesafenin hesaplanması işlemi yapılabilir. İşlem yapılırken aynı hesap makinesinde olduğu gibi fonksiyonlar iç içe kullanılmaktadır. Örneğin üst alma ifadesi $^$, karekök fonksiyonu içinde kullanılmaktadır (Şekil 17).



Şekil 17

Ondalık hane sayısı belirleme (Şekil 12)



Şekil 18

Ondalık hane sayısı belirlemek için **Round()** fonksiyonu kullanılır. Round() fonksiyonu içine iki değer alır. İlk değer yuvarlanacak olan sayı, ikinci değer ise ondalık hane sayısını ifade eden kalıp.

Şekil 18 örneğine göre Round() fonksiyonunun ikinci değeri:

1 değeri yazıldığında sonuç \rightarrow Round(sonuc, 1) \rightarrow 69

0.1 değeri yazıldığında sonuç \rightarrow Round(sonuc, 0.1) \rightarrow 69.1

0.01 değeri yazıldığında sonuç \rightarrow Round(sonuc, 0.01) \rightarrow 69.05

0.001 değeri yazıldığında sonuç \rightarrow Round(sonuc, 0.001) \rightarrow 69.051

Açı birimleri ve Trigonometrik fonksiyonlar

Derece, Radyan ve Grad olmak üzere üç farklı açı birimi mevcuttur. Kendi aralarında dönüşüm için:

ifade 3

$$\frac{R}{2\pi} = \frac{D}{360^\circ} = \frac{G}{400^g} \rightarrow \frac{R}{\pi} = \frac{D}{180^\circ} = \frac{G}{200^g}$$

ifade 3'de yer alan formüller kullanılır.

Programlama dillerinde trigonometrik fonksiyonlar sadece radyan değerleri kullanır. Örneğin derece açı biriminde olduğunu düşünerek 30° değerini sinüs trigonometrik fonksiyonuna aktarıp Mapbasic programında çalıştırdığımızda sonuç `sin()` Mapbasic programında sonucunu

Note `sin(30)` → program kodunun sonucu → -0.988031624 → değeridir

Programlama dillerinde trigonometrik fonksiyonlar radyan açı biriminde çalışırlar. Trigonometrik fonksiyon içine eklenecek açı değeri radyana dönüştürülerek eklenmelidir. Şekil 19 incelendiğinde derece açı birimindeki 18.457812° açısı Denklem 1'de yer alan formüllerden yararlanarak: $\sin(x * \pi/180)$ formülü kullanılır.

$$R = D * \frac{\pi}{180}$$

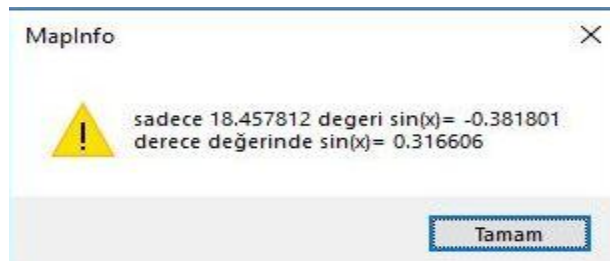
Formülün Mapbasic programlama dilinde yazılabilmesi için, π değeri için bir değişken tanımlanmalıdır. Çünkü Mapbasic programlama dili kütüphanesinde π değeri tanımlanmamıştır.

```

acidegerleriradyanderece.MB
dim x as float
dim pi as float
pi=3.14159265358979
'18.457812 değerinin derece açı biriminde olduğunu düşünelim
x=18.457812
note "sadece 18.457812 degeri sin(x)= " + sin(x) + chr$(10) + "derece değerinde sin(x)= " + sin(x*pi/180)

```

Şekil 19



Şekil 20

Şekil 19 içinde ki kod incelendiğinde `chr$(10)` kodu kullanılmıştır. `Chr$()` fonksiyonu içine sayısal değer alır. Bu sayısal değerler ASCII tablosunda (Tablo 2 Sayısal sütunları)

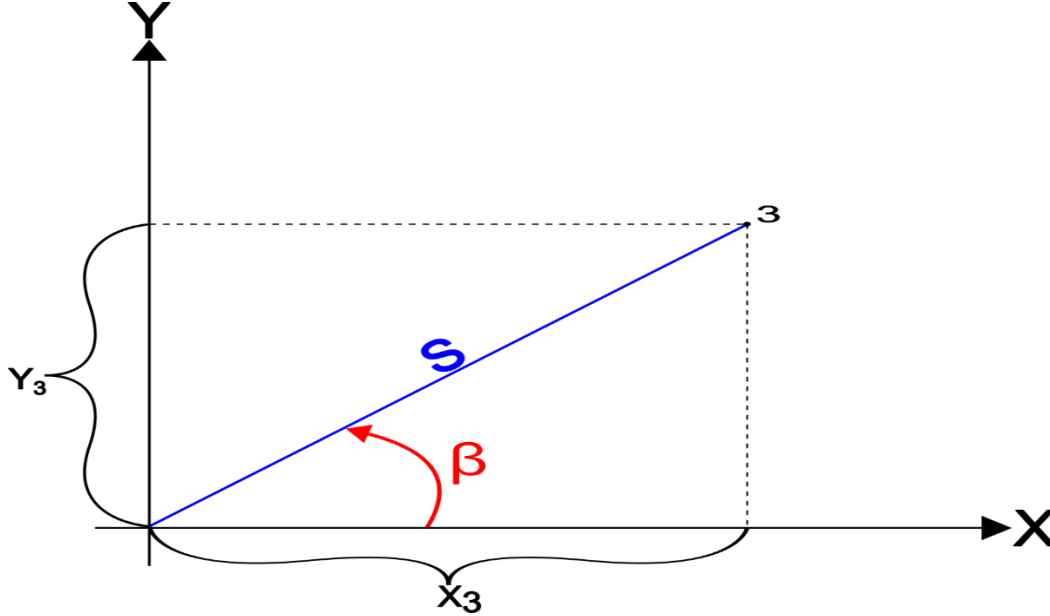
tanımlanmış sayısal değerlerdir. Sayısal değerler 0 ile 127 arasındadır. *Chr\$()* fonksiyonu içine eklenecek sayısal değerler sonucu fonksiyon geriye tanımlanmış sabit karakterler geriye döndürür. 10 sayısı ASCII tablosunda yeni satır anlamındadır. *Chr\$(37)* geriye % sembolünü geriye döndürür.

Tablo 2

Sayısal	Karakter	İşlem	Sayısal	Karakter	Sayısal	Karakter	Sayısal	Karakter
l	r		l	r	l	r	l	r
0	NUL	(null)	32	SPACE	64	@	96	`
1	SOH	(start of heading)	33	!	65	A	97	a
2	STX	(start of text)	34	"	66	B	98	b
3	ETX	(end of text)	35	#	67	C	99	c
4	EOT	(end of transmission)	36	\$	68	D	100	d
5	ENQ	(enquiry)	37	%	69	E	101	e
6	ACK	(acknowledge)	38	&	70	F	102	f
7	BEL	(bell)	39	'	71	G	103	g
8	BS	(backspace)	40	(72	H	104	h
9	TAB	(horizontal tab)	41)	73	I	105	i
10	LF	(NL line feed, new line)	42	*	74	J	106	j
11	VT	(vertical tab)	43	+	75	K	107	k
12	FF	(NP form feed, new page)	44	,	76	L	108	l
13	CR	(carriage return)	45	-	77	M	109	m
14	SO	(shift out)	46	.	78	N	110	n
15	SI	(shift in)	47	/	79	O	111	o
16	DLE	(data link escape)	48	0	80	P	112	p
17	DC1	(device control 1)	49	1	81	Q	113	q
18	DC2	(device control 2)	50	2	82	R	114	r
19	DC3	(device control 3)	51	3	83	S	115	s
20	DC4	(device control 4)	52	4	84	T	116	t
21	NAK	(negative acknowledge)	53	5	85	U	117	u
22	SYN	(synchronous idle)	54	6	86	V	118	v
23	ETB	(end of trans. block)	55	7	87	W	119	w
24	CAN	(cancel)	56	8	88	X	120	x
25	EM	(end of medium)	57	9	89	Y	121	y
26	SUB	(substitute)	58	:	90	Z	122	z
27	ESC	(escape)	59	;	91	[123	{
28	FS	(file separator)	60	<	92	\	124	
29	GS	(group separator)	61	=	93]	125	}
30	RS	(record separator)	62	>	94	^	126	~
31	US	(unit separator)	63	?	95	_	127	DEL

Mapbasic dilinde kullanılan trigonometrik fonksiyonlar:

Yabancı kökenli tüm CAD (computer aided design – bilgisayar destekli tasarım) ve GIS (Geographic Information System – coğrafi bilgi sistemleri) yazılımları yatay düzlem eksenlerini (X ve Y eksenini) Şekil 21’de olduğu gibi X eksenini sağa – doğuya doğru artacak şekilde, Y eksenini ise yukarı – kuzeye doğru artacak şekilde kabul ederler.



Şekil 21

Türkiye’de harita yapımı amaçlı kullanılan X – Y yatay düzleminde X eksenini yukarı doğru artar, Y eksenini ise doğuya doğru artar. Türkiye’de elde ettiğimiz X koordinat değerinin Mapinfo gibi yabancı kökenli yazılımlardaki karşılığı Y koordinatıdır. Türkiye’de elde ettiğimiz Y koordinat değerinin Mapinfo gibi yabancı kökenli yazılımlardaki karşılığı X koordinatıdır. Mapbasic derleyicisinde kullanacağımız yazılımda da bu şekilde kullanılmalıdır.

Tablo 3 Mapbasic derleyicisinde kullanılan programlama dilinde var olan trigonometrik fonksiyonların adları bulunmaktadır.

Tablo 3

Fonksiyon adı	Fonksiyon özelliği
$\cos(\beta) = X_3 \div S$	Cosinüs fonksiyonu
$\sin(\beta) = Y_3 \div S$	Sinüs fonksiyonu
$\tan(\beta) = Y_3 \div X_3$	tanjant fonksiyonu
$\text{acos}(X_3 \div S) = \beta$	arccosinüs fonksiyonu
$\text{asin}(Y_3 \div S) = \beta$	arcsinüs fonksiyonu

$$\text{atn}(Y_3 \div X_3) = \beta \quad \text{arctanjant fonksiyonu (Mapbasic Dilin Kullanımı)}$$



Programlama dillerinde trigonometrik fonksiyonlar, fonksiyon içine açı değeri alırken, açı değerinin açı birimi radyan olmalıdır. Yazılan kod içinde kullanılacak açı değeri derece veya grad olması gerekiyorsa, açı değeri radyana dönüştürülmelidir.

ifade 4 cosinüs fonksiyonun program içinde gösterimi bulunmaktadır. α açısı grad açı birimindedir. Program içinde α açısını cosinüs fonksiyonu içinde kullanmak istiyorsak ilk önce açıyı radyan açı birimine dönüştürmemiz gerekecektir. Bu işlemi yapmak için cosinüs fonksiyonu içinde α ilk önce π (pi) ile çarpılıp 200^g değerine bölünmelidir.

ifade 4

$$\cos(\beta * \pi \div 200^g)$$

Arctanjant fonksiyonu kendi içerisine uzunluk değerlerini alır geriye açı biriminde sonuç döndürür. Tüm programlama dillerinde açı birimi radyandır. Arctanjant fonksiyonunun döndüreceği açı değerini hangi açı biriminde görmek istiyorsak ifade 5 örneğinde olduğu gibi sonuç açı dönüştürülmelidir. ifade 5 işleminde sonuç β açı değeri radyandır. Bu değeri grad cinsinde görmek istiyorsak 200^g ile çarpıp π (pi) değerine bölmeliyiz.

ifade 5

$$\text{atn}(Y_3 \div X_3) = \beta * 200^g \div \pi$$

Şekil 22 cosinüs fonksiyonunun kullanımını ve üç açı biriminde de nasıl kullanılacağına bir örnektir.

```

acidegerleriradyanderecegrad.mb
dim x,pi as float
pi=3.14159265358979
x=35.78954754
note "x= "+ x + chr$(10)
+"radyan cinsinde cos(x)= "+cos(x)+chr$(10)
+"derece cinsinde cos(x)= "+cos(x*pi/180)+chr$(10)
+"grad cinsinde cos(x)= "+cos(x*pi/200)

```

MapInfo

x= 35.7895
radyan cinsinde cos(x)= -0.332325
derece cinsinde cos(x)= 0.811171
grad cinsinde cos(x)= 0.846095

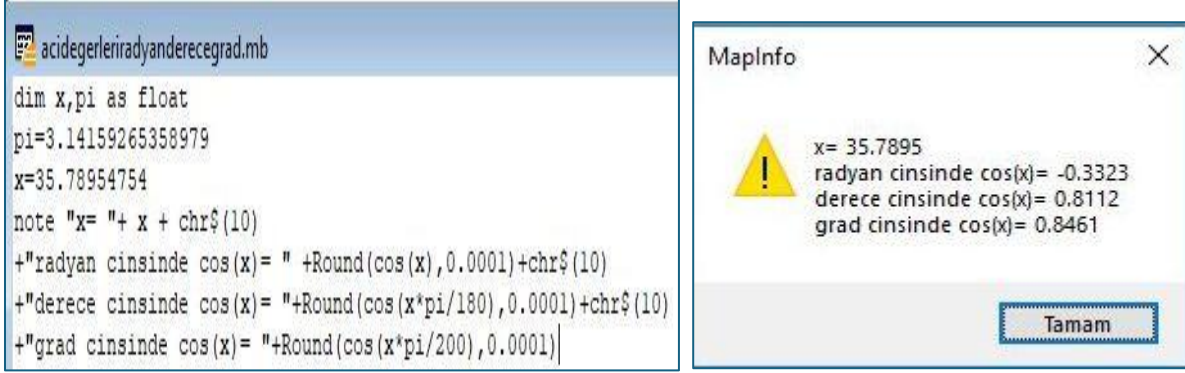
Tamam

Şekil 22

Şekil 22 soldaki kod ve sağdaki sonuç resimleri incelendiğinde, kod içinde Note komutunda uzun bir cümle yazılmış fakat sonuç çıktının 4 satırdan oluşması sağlanmıştır. Bunu yapabilmek için chr\$(10) ifadesi kullanılmıştır. chr\$(10) ifadesi Enter tuşunun yaptığı işlemi yapacaktır. chr\$(10) ifadesinden sonra gelen " " aralığındaki cümle bir alt satıra yazılacaktır.

İç içe fonksiyonların kullanımı

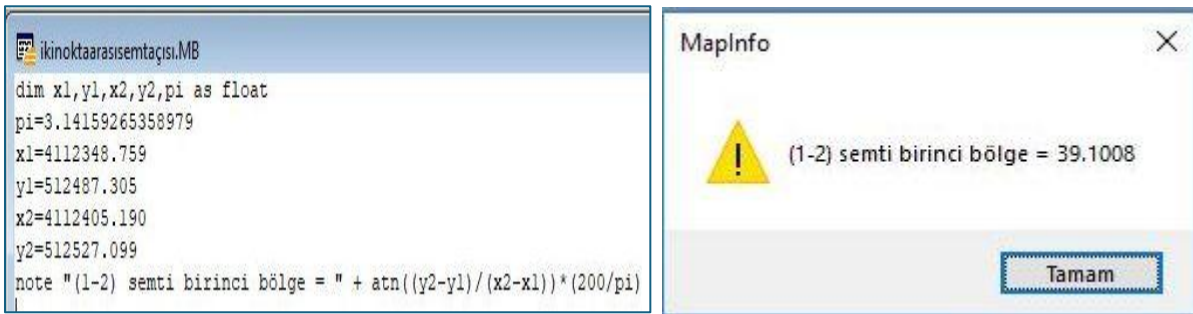
Birden fazla farklı fonksiyon iç içe kullanılabilir. Örneğin trigonometrik fonksiyonların sonuç değerlerinin ondalık hane sayısının 4 olması için Round() fonksiyonu içinde cosinüs() fonksiyonu kullanılabilir. Şekil 16 işlemin nasıl yapıldığına bir örnektir. Şekil 15’de yapılan örnek üzerinde değişiklik yapılarak sonuç elde edilmiştir.



Şekil 23

Semt açısının Hesaplanması

Semt açısının hesaplanması için arctanjant fonksiyonu yani Mapbasic’de atn() fonksiyonu kullanılacak. Şekil 24’de Atn() fonksiyonu dikkatli incelendiğinde önceki trigonometrik fonksiyonlardan farklı olarak dönüşüm işlemi fonksiyonun dışında olmuştur ve Denklem 1’de ki formül işlemde kullanılmıştır. Ters trigonometrik fonksiyonlar içine uzunluk alırlar ve geriye açı döndürürler. Mapbasic programlama dilinde Ters trigonometrik fonksiyonlar geriye radyan cinsinde açı değeri döndürür. Şekil 24’de sonucun grad açı biriminde bulunması için atn() fonksiyonu (200/pi) değeriyle çarpılmıştır.



Şekil 24

Karşılaştırma Yapıları

Semt açısı (Şekil 24) örneğinde semt açısı birinci bölgede hesaplanmıştır. Semt açısını bölgesi bilindiği için, semt açısını bölgesi araştırılmadan, kolaylıkla hesaplanmıştır. Örnekte nokta koordinatları program kodu içinde girildi. Eğer noktaların koordinatları kullanıcıdan alınsaydı, o takdirde program kodu içinde semt açısının bölgesinin belirlenmesi ve semt açısının

belirlenen bölgeye göre hesaplanması gerekirdi. Semt açısının hesaplanmasına dair konu anlatımı Ekler kısmında Ek-1’de verilmiştir.

Semt açısı gibi hesaplamaların yapılması için karşılaştırma (sınama) yapılarının kullanılması gerekir.

İf – Then Karşılaştırma Yapısı

If kelimesinin Türkçe karşılığı *Eğer* kelimesidir. If – Then sorgulama yapısında, yapılacak bir karşılaştırmanın olumlu olması halinde yapılacak işlemler kod bloğunun olumlu olduğu tarafta olur, karşılaştırmanın olumsuz olması halinde yapılacaklar kod bloğunun olumsuz kısmı içinde belirtilir. Tablo 4 If-then yapısının kullanımına bir örnektir.

Tablo 4

If (karşılaştırma 1) Then
(karşılaştırmanın Olumlu olduğu kısım) Yapılacaklar 1
Else
(karşılaştırmanın Olumsuz olduğu kısım) Yapılacaklar 2
End If

Tablo 4 içinde yazılan if – Then bloğunda anlatılmak istenen:

- **Eğer karşılaştırma 1 doğruysa** *Yapılacaklar 1* kod bloğu uygulansın,
- **Eğer karşılaştırma 1 doğru değilse** *Yapılacaklar 2* kod bloğu uygulansın.

Else Türkçe karşılığı *değilse* anlamına gelir. Karşılaştırmanın doğru olmadığı hallerde yapılacaklar için ek bir blok olmasını sağlar.

Karşılaştırma işlemlerinin yapılabilmesi için Karşılaştırma Operatörlerine ihtiyaç vardır. Eğer birden fazla karşılaştırma yapılacaksa, mantıksal operatörlere ihtiyaç vardır.

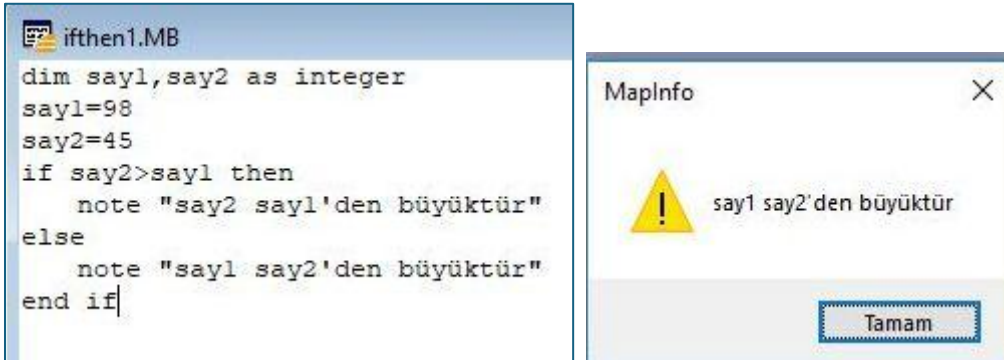
Karşılaştırma Operatörleri:

Tablo 5 Mapbasic programlama dilinde, karşılaştırma amaçlı kullanılan karşılaştırma operatörlerinin listesidir. If – Then yapısında karşılaştırma işlemleri için bu operatörler kullanılacaktır.

Tablo 5

Operatör	örnek	Eğer doğruysa
=	a = b	a, b'ye eşittir
<>	a <> b	a, b'ye eşit değildir
<	a < b	a, b'den küçüktür.
>	a > b	a, b'den büyüktür.
<=	a<=b	a, b'den küçük veya eşittir.
>=	a>=b	a, b'den büyük veya eşittir.

Şekil 25 iki sayının karşılaştırılması için kullanılan programı ifade eder. Program kodunun anlatmak istediği: **eğer** say2 say1'den büyükse ekrana mesaj kutusu çıkar ve say2 say1'den büyüktür yaz, **değilse** ekrana mesaj kutusu çıkar ve say1 say2'den büyüktür yaz

**Şekil 25**

Şekil 25 örneği incelendiğinde if – then yapısı End If koduyla sonlanıyor.

Mantıksal Operatörler:

İki değeri karşılaştırmak için Şekil 25 örneğinin yapısı çok uygundur. Fakat ikiden fazla sınımayı (karşılaştırmayı) **aynı anda** uygulamak gerektiği takdirde mantıksal operatörler kullanılır. Tablo 6'de A bir karşılaştırmayı, B ayrı bir karşılaştırmayı göstermektedir.

Tablo 6

Mantıksal Operatör	örnek	Açıklama	Uygulamada
And	A and B	A ve B	Ya her iki sınama da doğru ya da her iki sınama da yanlış olması halinde uygulanır
Or	A or B	A veya B	İki sınımadan biri dahi doğru ise uygulanır
Not	Not A	A değil	A sınavasının olmadığı durum uygulanır

ElseIf Yapısı

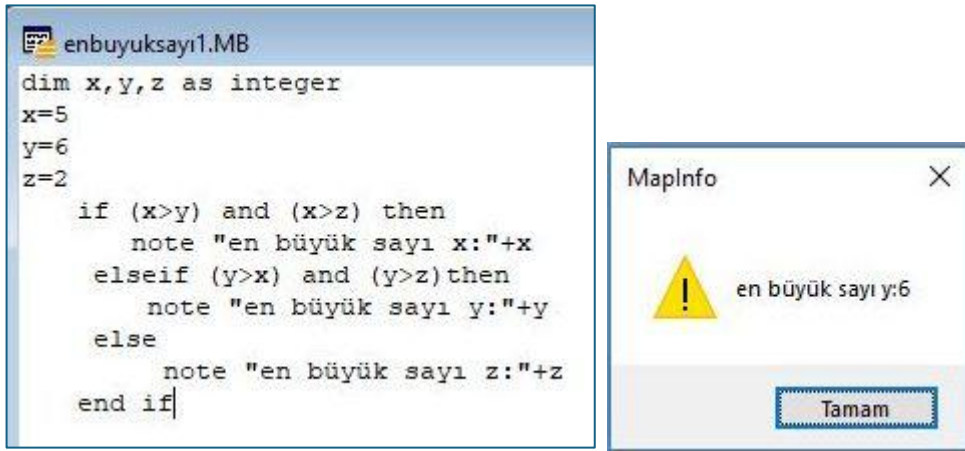
Şekil 26 verilen üç sayı içinden en büyük sayıyı bulduran algoritmadır. Örnek içinde **ElseIf** komutu kullanılmıştır. Bu komutu çalışma mantığı örneğe göre incelenirse:

Eğer x, y'den **ve** z'den büyük ise x en büyüktür diye ekrana yazılacak. Değilse tekrar sınama yapabilmek için **Elseif** yapısı kullanılarak yeni bir sınama yapılacaktır. Yazılan kodu tekrar baştan okunduğunda:

Eğer x, y'den büyük **ve** x, z'den büyük ise ekrana en büyük sayı x:5 yaz.

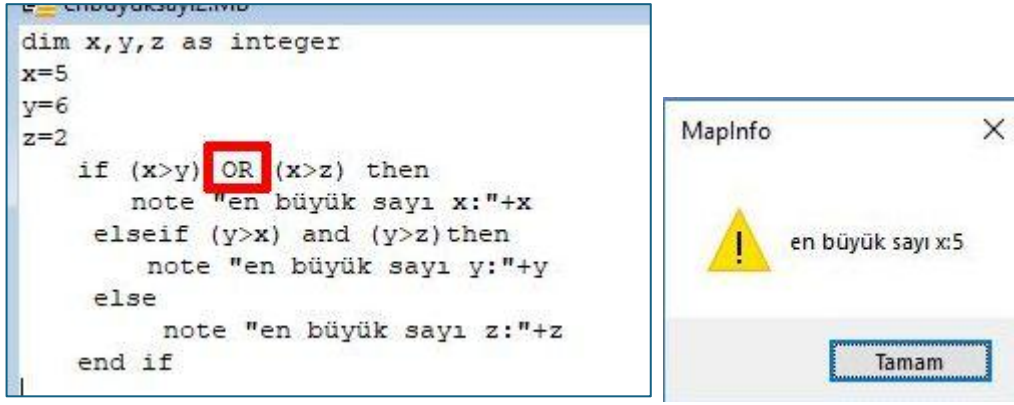
Eğer Değilse y x'den büyük **ve** y z'den büyük ise ekrana en büyük sayı y:6 yaz.

Değilse ekrana en büyük sayı z:2 yaz.



Şekil 26

Şekil 27 OR mantıksal operatörüne örnektir. Bir önceki örnekte olduğu gibi verilen üç sayı içinden en büyük olanın bulunması için kod yazılmıştır. Kod içinde or kullanılmıştır. OR operatörü, yapılan iki sınımadan biri dahi doğru olsa işlemin yapılmasını sağlar. Program kodunda ilk if sınavasında X Y'den büyükse (5 6'dan büyükse) **veya** X Z'den büyükse (5 2'den büyükse) ekrana X büyük yazacaktır. OR operatörünün doğru çalışması için iki sınımadan sadece birinin doğru olması yeterlidir. Ama AND operatöründe ise iki sınımanın da doğru olması gerekir. OR operatörü bu kodda yanlış kullanılmıştır.



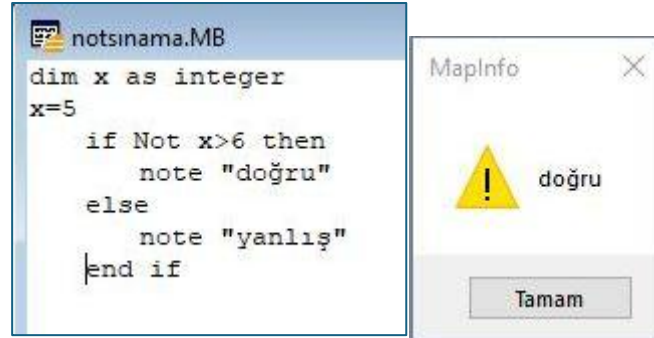
Şekil 27

Not operatörünün kullanımı

Şekil 21 Not mantıksal operatörüne basit bir örnektir. Program kodu:

Eğer x değeri 6'dan büyük **DEĞİLSE** ekrana doğru yaz
karşılaştırma doğru değilse ekrana yanlış yaz

Not kullanımında iki karşılaştırma yok sadece bir karşılaştırmanın olmadığı durumda karşılaştırmayı doğru kabul eder. Örnekte de 5 6'dan büyük olmadığı için karşılaştırma Not operatörü sebebiyle doğru kabul edilir.



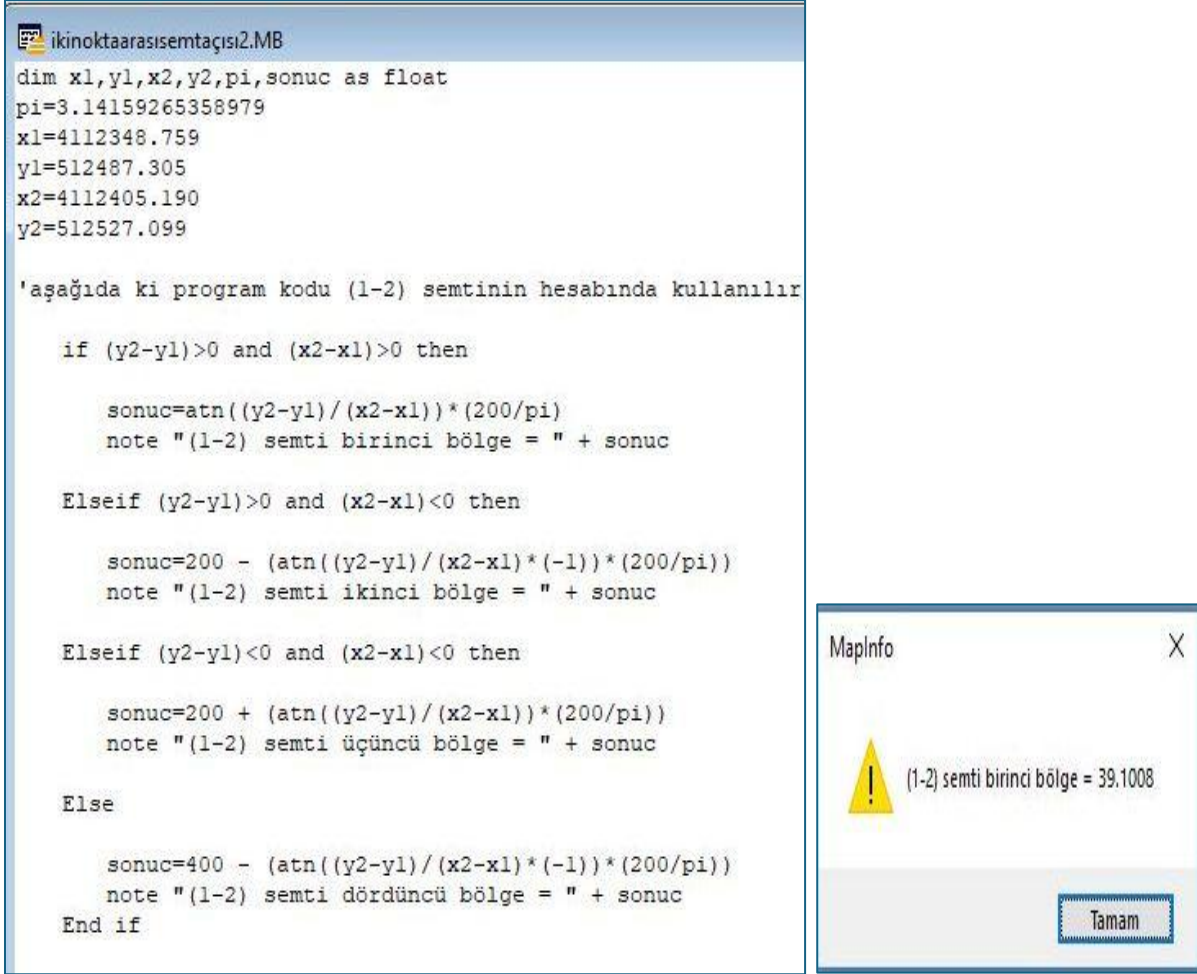
Şekil 28



If – then yapısında, else veya elseif yapıları kullanılacaksa, ilk sınama doğru olduğu takdirde kod else veya elseif yapısı içine uğramaz. O yüzden kod algoritması düzgün kurulmalıdır.

İki nokta arasındaki Semt Açısının Hesaplanması

Şekil 29 sol resim koordinatları verilen iki nokta arasındaki semt açısının bulunmasına dair program kodunu içermektedir. Program kodunda kullanılan iç içe if – then yapısı sayesinde istenilen semtin bölgesinin bilinmesine gerek kalmadan, programın bölgeyi tespit edip bölgeye göre semt açısını bulması sağlanmıştır.



Şekil 29

Procedur (Yordam) ve Function (Fonksiyon) Kavramı

Program kodu yazılırken, programın daha etkin olmasının sağlanması ve tekrara yer vermeyerek yazılması önemlidir. Aynı işleme ait kod satırları, program kodu içinde birden fazla defa tekrarlanıyorsa, tekrarlanan kod satırları Yordam ve/veya Fonksiyon içine yazılarak programa etkinlik kazandırılır. Tekrarlanan kod satırları, yordam veya fonksiyon içinde tanımlandığında, kod satırları tekrar yazılmaya gerek kalmadan sadece tanımlandığı yordam veya fonksiyonun çağırılması yeterli olacaktır.

Fonksiyon ve yordam yapıları kendi değişkenlerini barındıran kod bloklarıdır. Kendi aralarındaki temel fark ise fonksiyon çağrıldığı kod satırına **geri değer döndürür**, yordam ise çağrıldığı kod satırına **geriye değer döndürmez**. Örneğin hesap makinesinde kullandığımız karekök, trigonometrik fonksiyonlar birer fonksiyondur. Trigonometrik fonksiyonlar içerisine değer alır, geriye değer döndürürler.

Yordam Tanımlanması ve Çalışma Prensibi

Program kodu yazılırken bir tasarım yapılmalıdır. Programı kullanacak kişinin yapacağı her bir işlemin o an bir **olay** olacağı unutulmamalıdır. Düğmeye basılması, onay kutusunun onaylanması, verilerin dosyadan çekilmesi, verilerin listelenmesi, verilerin sıraya konulması, kullanıcının kutuya yazı yazması, rakam yerine harf yazılması gibi işlemler bir olaydır. Yordam tekrarlı yapılacak olaylara müdahale edilmesi işlemlerinde kullanılan kod parçalarıdır. Programı kullanacak kişinin, programın farklı aşamalarında kontrol edilmesi gerekebilir, programın farklı aşamalarında kullanacağı düğmelerde veya metin kutularında yönlendirme yapılması gerekebilir. Yordam devamlı yapılacak denetim işlemlerinde ve yönlendirme işlemlerinde kullanılır. Her defasında aynı kontrol işlemi için aynı kod satırları yazılacağına, yordam içinde kod satırları bir kere yazılır, her defasında yordamın iş yapması için yordam **çağırılır**.

Program içinde kullanılacak bir yordam önceden deklare edilmeli (bildirilmeli) daha sonra da program kod satırı olarak yazılmalıdır.

declare sub yordam_adi → kod satırında declare kelimesi yordamın kod içinde olacağı bildiriliyor. Yordamın bildirilmesi, program kod bloğunun başlangıcında yapılmalıdır.

Yordam kod bloğu yazılırken sub komutu ile başlar ve end sub ile yordamın bittiği belirtilir.

Sub yordam_adi

Yordam içinde yazılacak kod satırlarını olduğu kısım

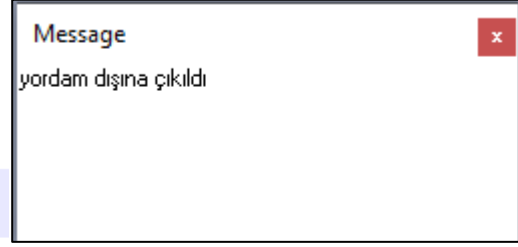
End sub

Yordam kod bloğunun çalışması için yordam program kod satırları içinde çağırılmalıdır. Şekil 30 bir yordam örneğidir. Şekil 30 sol resim 1 numaralı satırda yordamın bildirimini yapmıştır. 2 numaralı satırda yordam kod bloğu *sub printyaz* satırı ile başlamış ve 4 numaralı satırda *end sub* satırı ile yordam kod bloğu bitirilmiştir. Program çalıştırıldığında, sonuç Şekil 30 olduğu gibi Mapinfo programında gözükecektir. Sonuç çıktı ekranı incelenirse, program kodu içinde yordam hiç çalışmadan, yordam sonrasındaki kod satırı çalışmıştır. Bunun nedeni, yordam çağırılmamıştır.

```

1 declare sub printyaz
2 sub printyaz
3   print "yordam çalıştı"
4 end sub
5   print "yordam dışına çıkıldı"

```



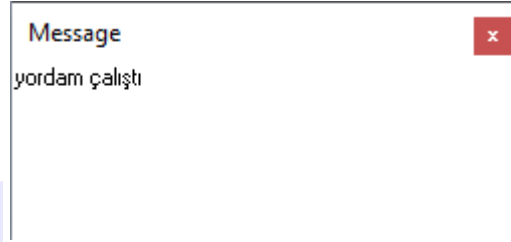
Şekil 30

Şekil 31 sol resimdeki kod incelendiğinde, Şekil 30 sol resimdeki kod satırından farklı olarak 2. Satırda *call* komutu ile yordam çağırılmıştır. Şekil 31 sağ resim, program kod bloğu çalıştırıldığında elde edilen sonuç ekran görüntüsüdür. *Call* komutu ile yordam çağırılmış ve *printyaz* isimli yordam bloğu içindeki kodlar çalışmıştır.

```

1 declare sub printyaz
2 call printyaz
3 sub printyaz
4   print "yordam çalıştı"
5 end sub

```



Şekil 31

Program Yönetiminde Ana Yordamın Kullanılması

Ana yordam programın yönetilmesi için kullanılan yordamdır. Tüm program bu yordam içinden yürütülür. Tasarımda bu yordama göre yapılır. Ana yordam kullanıcı tarafından tanımlanan bir yordam değildir. Ana yordam MapBasic kütüphanesi içinde sabit bir isim ile tanımlanmıştır. Ana yordam ismi **Main** olarak ifade edilir. Ana yordam da diğer yordamlar olarak tanımlanmak zorundadır. ifade 6 Main tanımlanması ve kullanımı örneğidir.

ifade 6

```

declare sub main
sub main
(ana yordam içinde
tanımlanacak kod bloğu)
end sub

```

Eğer Main tanımı yapılmış ve program içinde kullanılmışsa, program ilk çalıştırıldığında direkt olarak main yordamı içerisine girer. Program Main yordamı tanımı gördüğü anda, Main yordamı öncesi ve sonrasında bir yordam veya fonksiyon içinde

yazılmamış kendi başına yazılmış kod satırlarını dikkate almaz. Eğer main yordamı dışında kalan, herhangi bir yordam veya fonksiyon dışında kalmış satırlar varsa program derlendiğinde hata mesajı ortaya çıkar.

Şekil 32 Main kullanımını ve main dışında kalmış, herhangi bir yordam veya fonksiyon içinde kullanılmamış, bir kod satırı örneği vardır. 2. Satırda *print* kodu kullanımı vardır.

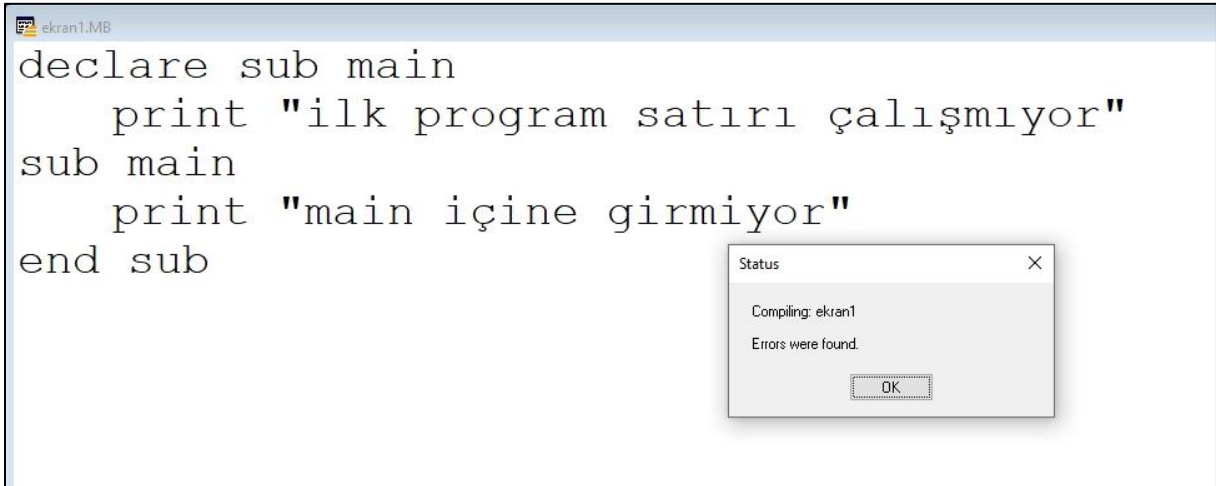
```

1 declare sub main
2     print "ilk program satırı çalışmıyor"
3 sub main
4     print "main içine girmiyor"
5 end sub

```

Şekil 32

Program derlendiğinde ve main yordamı tanımını gördüğü anda main yordamı içine girecektir. Program main yordamı üzerinden programın yönetileceğini düşünür. Main yordamı dışında tek başına yazılmış (bir fonksiyon veya yordam içinde olmayan, Şekil 32 2. satır) bir satır gördüğünde hata verir. Şekil 32 örneğindeki program kodu derlendiğinde, program 2 .satırı görür ve Şekil 33 resminde görülen hatayı verir. “Ok” düğmesine basıldığında Şekil 34’de görülen hata penceresi açılır.



Şekil 33

[ekran1.mb:2] print is not valid outside a Sub procedure or Function.

Şekil 34

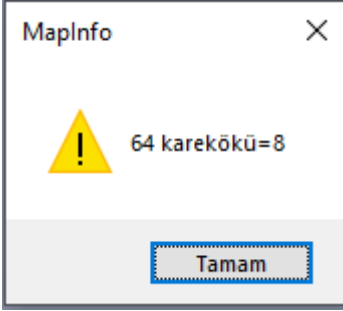
Karekök Yordam Örneği

Şekil 35 64 sayısının karekök değerini bulan yordam tanımı vardır. *karekok* adlı yordam 1. Satırda tanımlanmıştır (Şekil 35). 2. Satırda yordam çağırılmıştır. 3. Satır ile 7. Satır arası *karekok* yordamı kod bloğudur. Yordam içinde *sayi* adlı tamsayı veri tipinde bir değişken tanımlanmış ve 64 sayısı değişkene aktarılmıştır.

```

1 declare sub karekok
2 call karekok
3 sub karekok
4     dim sayi as integer
5     sayi=64
6     note "64 karekökü="+sqr(sayi)
7 end sub

```



Şekil 35

İçine Değer Alan (Parametre alan) Yordam Tanımlanması

Yordam ve/veya fonksiyon, program kod bütünü içinde tekrarlanan kod satılarının daha etkin bir şekilde tekrarını önlemek için kullanılan yapılardır. Şekil 35 örneğinde, yordamın çağırıldığı yerde yordam içinde kullanılmak üzere, yordama değer ataması yapılmamıştır. Eğer yordamın çağırıldığı yerde yordam içinde kullanılmak üzere parametre ataması yapılması gerekiyor ise yordam tanımı farklılaşır

Sub yordam_degersiz

Kod satırları

End Sub

Sub yordam_degerli(Byval parametre as veritipi)

Kod satırları

End Sub

Yukarıda yordam tanımlama örnekleri vardır. Solda içerisine değer/parametre almadan tanımlanmış yordam bulunmaktadır. Sağda ki yordam tanımında içerisine değer/parametre alan bir yordam tanım bloğu bulunmaktadır.

Eğer yordamın çağırıldığı satırda, yordam içinde kullanılmak üzere değer/parametre eklenmesi gerekiyorsa, yordam tanımında parantez içinde eklenecek değer/parametre tanımlanmış bir yordam tanımlanması gerekir.

Açı Değerini Derece – Dakika – Saniye Açılış Birimine Dönüştürüp Ekranaya Yazan Yordam

Şekil 36 rasyonel sayıda verilen derece açı birimindeki değeri ekrana derece – dakika – saniye şeklinde yazan yordama ait kodu göstermektedir. Kod incelendiğinde yordam 6. ile 15. satırlar arasındadır. Yordam içinde tanımlanan değişkenler (Şekil 36 örneğinde tamsayı olarak tanımlanan derece, dakika değişkenleri ve rasyonel sayı olarak tanımlanan saniye değişkenleri) sadece tanımlandığı yordam içinde geçerlidir.

Şekil 36 1. Satırda yordam tanımı yapılıyor. Tanımlanan *dds_ekranyaz* adlı yordam tanımında, yordamın *float* veri tipinde bir değer alacağı belirtilmiş. 5. Satırda yordam çağırılırken,

→call dds_ekranyaz(değer)

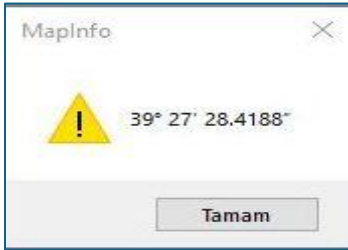
Call komutu yordamı çağırırken, parantez içine eklenen değer/değişken yordam içinde kullanılacaktır. Şekil 36 10., 11. ve 12. Satırlarda yordama eklenen değer/parametrenin kullanımı görülmektedir.

```

yordam_dds.MB
1  declare sub dds_ekranyaz(byval veri as float)
2
3  dim deger as float
4  deger=39.45789412
5  call dds_ekranyaz(deger)
6  sub dds_ekranyaz(byval veri as float)
7      dim derece,dakika as integer
8      dim saniye as float
9
10     derece=fix(veri)
11     dakika=fix((veri-derece)*60)
12     saniye=((veri-derece)*60-dakika)*60
13     note derece + chr$(176) + " " + dakika + chr$(145) + " " + saniye + chr$(147)
14
15 end sub

```

Şekil 36



Şekil 37

Fonksiyon Tanımlanması ve Fonksiyon Kullanımı Örneği

Fonksiyon, program kod bütünü içerisinde birden fazla kez kullanılacak kod bloklarının tekrarını önlemek, programın daha etkin ve hızlı olmasını sağlayacak yapılardır. Yordam yapıları da bu işlevi yerine getirirler. Fonksiyon yapısının çalışması sonucu geriye bir değer döndürür. Yordam geriye değer döndürmez. Matematikte ve geometride kullandığımız fonksiyon yapıları ile aynı mantıkta çalışır. Fonksiyon kavramını anlamak için trigonometrik fonksiyonları düşünebiliriz. Trigonometrik fonksiyonlar (sinüs, cosinüs, tanjant fonksiyonları) içlerine açı biriminde değer alır geriye birimsiz değer döndürürler. Daha önce anlatılan yordam kavramında hem yordamın içine parametre almadan hem de içerisine parametre olarak kullanımları gösterilmiştir.

Basic programlama dilinin bir türevi olan Mapbasic programlama dilinde de fonksiyon tanımı yapılırken, aynı matematikte kullandığımız fonksiyonlar gibi, fonksiyon içerisine parametre alır. Fonksiyon tanımı yapılırken, içine alacağı parametrenin (veya parametrelerin) ad tanımı yapılmalı, parametrenin değişken tipi belirtilmelidir. Ayrıca fonksiyon geriye değer döndüreceği için, geriye döndüreceği verinin değişken veri tipi de belirtilmelidir.

(1) *Declare function fonksiyon_adi(Byval parametre1 as degisken_tipi,.Byval parametre2 as degisken_tipi,...) as degisken_tipi*

(2) *function fonksiyon_adi(Byval parametre1 as degisken_tipi,.Byval parametre2 as degisken_tipi,...) as degisken_tipi*

(3) *fonksiyon_adi=geri_donecek_deger*

End function

Yukarıda (1) numarası ile belirtilen satırda fonksiyon bildirimini gözükmektedir. Yordam tanımında olduğu gibi ilk önce *declare* komutu kullanılmalıdır. İkinci komut olarak *function* komutu ile fonksiyon tanımı yapıldığı belirtilir. Parantez içinde fonksiyona aktarılacak değişkenler/parametrelerin adı ve veri tipi, parantezden sonra ise fonksiyonun geriye döneceği değerin veri tipi tanımı yapılmıştır.

Yukarıda (2) numarası ile belirtilen kod satırında fonksiyon tanımı yapılmaya başlamış ve *End Function* satırı ile fonksiyon bitirilmiştir

Yukarıda (3) numarası ile belirtilen kod satırında ise fonksiyon kod bloğunda fonksiyon geri dönüşü yaptığı kod satırının temsili vardır. Fonksiyon çağırıldığı yere, tanımında belirtildiği veri tipi ile geri dönüş yapacaktır.

Sayının Kuvvetini Alan Fonksiyon Örneği

Şekil 38 Fonksiyon tanımına bir örnektir. Örnekte, fonksiyon içine iki adet değişken almaktadır. İçine aldığı ilk değişken parametresi *float* veri tipindedir. İkinci değişken parametresi *integer* veri tipindedir. Fonksiyon geriye float veri tipinde değer döndürmektedir.

1. satırda fonksiyon bildirim yapılmıştır. 2. Satırda fonksiyon,

degeral(4.2,2)

komut satırı ile çağırılmıştır. *degeral* ismindeki fonksiyonun çağırılmasında 4.2 değerinin 2. Derece kuvvetinin bulunması istenmiştir.

4. satırda fonksiyon geriye değer döndürmektedir.

degeral=say^kuvvet

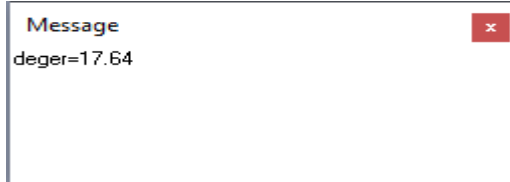
komutu 2. Satırda çağırıldığı yere geriye değer döndürür.

```

1 declare function degeral(byval say as float, byval kuvvet as integer) as float
2 print "deger="+degeral(4.2,2)
3 function degeral(byval say as float, byval kuvvet as integer) as float
4     degeral=say^kuvvet
5 end function

```

Şekil 38



Şekil 39

Açı Değerini Derece – Dakika – Saniye Açı Birimine Dönüştürüp Ekranaya Yazan Fonksiyon

Şekil 40 bir önceki yordam örneğinin (Şekil 36) fonksiyon halidir. Şekil 40 1. satırı fonksiyonu programa tanıttığımız kod satırıdır. Satırda yazılmış kod, *dds_ekranyaz* fonksiyonunun, içerisine float veri tipinde değer alacağını ve geriye *string* veri tipinde geriye değer döndüreceğini belirtir. Kod içinde 7. ile 17. Satırlar arasında fonksiyonun tanımı yapılmıştır. Fonksiyon tanımı *Function – End Function* kod bloğu arasındaki yapıdır. Fonksiyonun yordamdan ilk farkını 7. Satırın en sonundaki *as string* koduyla anlayabiliriz. Bu kodun anlamı fonksiyonda elde edilen değer, programı kod satırları içinde çağrıldığı yere (5. satır) *string* tipinde veri geriye döndüreceğidir. Geri dönecek veri tipi her yazılacak fonksiyonda değişebilir. Kod içinde fonksiyon 5. Satırda çağrılmıştır. **Yordamdan farklı olarak fonksiyon çağrılırken *call* kodu kullanılmasına gerek yoktur.**



Şekil 40 16. Satır fonksiyonun geriye veri döndürüldüğü kod satırıdır. Kod yazılırken,

dds_ekranyaz=tum

dds_ekranyaz fonksiyonun adıdır. *tum* değişkenindeki değer *dds_ekranyaz* fonksiyon ismine aktarılarak, geri dönüş işlemi yapılmış olur. Fonksiyon çağrıldığı 5. Satıra geriye veri döndürmüştür.



Fonksiyon 5. Satıra veriyi geri döndürdüğünde, program kaldığı yerden devam devam edecektir.

```

fonksiyon_dds.mb x
1 declare function dds_ekranyaz (byval veri as float) as string
2
3 dim deger as float
4 deger=39.45789412
5 note dds_ekranyaz(deger)
6
7 function dds_ekranyaz (byval veri as float) as string
8     dim derece,dakika as integer
9     dim saniye as float
10    dim tum as string
11
12    derece=fix(veri)
13    dakika=fix((veri-derece)*60)
14    saniye=((veri-derece)*60-dakika)*60
15    tum=derece+ chr$(176) + " " + dakika +chr$(145)+ " " + saniye +chr$(147)
16    dds_ekranyaz=tum
17 end function

```

Şekil 40



Şekil 41

İki Nokta Arasındaki Yatay Mesafenin Hesabını Yapan Fonksiyon

Şekil 42 örneği daha önce yapılan iki nokta arasındaki Kartezyen yatay mesafe değerinin fonksiyon ile bulunmasını göstermektedir. Şekil 42 1. Satırda *uzunluk* isiminde fonksiyon tanımı yapılmıştır. Fonksiyon içerisine birden fazla değişken eklenecek şekilde tanımlama yapılmıştır. Örnekte, fonksiyonun içerisine dört adet *float* veri tipinde değişken ekleneceği belirtilmiştir. Şekil 40 örneğinde fonksiyonun geri dönüş değeri string veri tipindeyken, Şekil 42 örneğinde ise geri dönen değer float olarak belirtilmiştir. Çünkü fonksiyon içinde bulunacak ve geriye döndürülecek olan yatay mesafe değeri reel sayıdır.

```

fonksiyon_uzunluk.mb
1  declare function uzunluk(Byval x1,y1,x2,y2 as float) as float
2
3  dim xdeg1,ydeg1,xdeg2,ydeg2 as float
4
5  xdeg1=4208135.442
6  ydeg1=477231.086
7  xdeg2=4208224.773
8  ydeg2=477286.399
9
10 note uzunluk(xdeg1,ydeg1,xdeg2,ydeg2)
11
12 function uzunluk(Byval x1,y1,x2,y2 as float) as float
13
14 uzunluk=sqr((y2-y1)^2+(X2-x1)^2)
15
16 end function
17
MapInfo
! 105.069
Tamam

```

Şekil 42

Form Tasarımı ve Kullanıcı Ara Yüzüyle Programlama

Form program kullanıcısının veri girebildiği, seçim yapabildiği, onay verebildiği araçların olduğu ara yüzlerdir. Veri girişi, seçim işlemi, onay işlemi gibi işlemlerin yapılabilmesi için araçlar vardır. Form bahsedilen araçların yerleştirildiği bir yüzeydir. Araçların her biri nesnedir ve her biri üzerinde işlem yapıldığında bir olay oluşur. Form üzerine araçlar yerleştirilirken, formu kullanacak kişiye veya araçların sahip oldukları olaylara göre tasarım yapılır. Form tasarımında kullanılan araçlar konu içinde örnekler yardımıyla tanıtılacaktır.

MapBasic Geliştirme Ortamı MapBasic IDE

MapBasic Programı, basit programların yazılıp, derlenmesi ve çalıştırılması açısından kullanışlı olabilir. Ama görsel bir form tasarlanması, diyalog kontrollerinin yönetilmesi açısından kullanışlı bir editör değildir. Dialog kontrollerinin form üzerindeki yerlerinin belirlenmesi, kontrollerin özelliklerinin belirlenmesi işlemlerinin Mapbasic programlama diliyle yapılması zahmetlidir.

Algoritma ve Bilgisayar Programlama dersinde MapBasic IDE programının kullanımın esas amacı, MapBasic programlama diliyle form ve dialog tasarım araçlarını öğrenmek ve geliştirmek için gereken uygulama ve geliştirme zamanının dersin saatlerince karşılanamayacağıdır. MapBasic IDE programının görsel form geliştirme ortamı sayesinde

form tasarımı işlemleri daha kolay öğrenilecek, esas öğrenilmesi gereken form araçlarıyla olay yönelimli programlama kavramına odaklanılması sağlanacak.

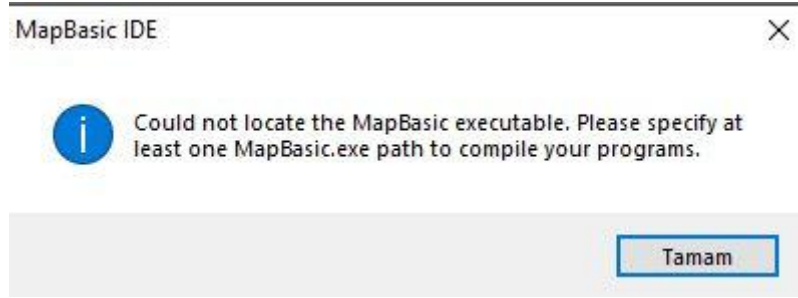
MapBasic IDE programının görsel form tasarımı yapılacak, yine MapBasic IDE programının bir aracı sayesinde geliştirilen form kolaylıkla MapBasic programlama diline dönüştürülecek. Dönüştürülen program kodu kopyalanıp, MapBasic editörüne yapıştırılacak. Yazılacak programın geriye kalan kısmı MapBasic editöründe tamamlanacak. Bu işlemin yapılmasının temel nedeni, MapBasic IDE programı farklı bilgisayarlarda farklı hataların vermesidir. Sınıf içindeki dersler homojen bir öğretimin sağlanabilmesi için MapBasic IDE yazılımı sadece kolay bir şekilde form tasarımının yapılması amacı için kullanılacaktır.

MapBasic IDE Programının Bilgisayara Kurulması İşlem Adımları

Mustafa ÖZÇETİN'in hazırladığı MapBasic Geliştirme Ortamı MapBasic IDE, Dialog kontrolleri yardımıyla form tasarımının görsel olarak yapılmasını sağlayan bir editördür. Program Mustafa ÖZÇETİN'e ait internet sayfası üzerinden indirilebilir (link aşağıda verilmiştir).

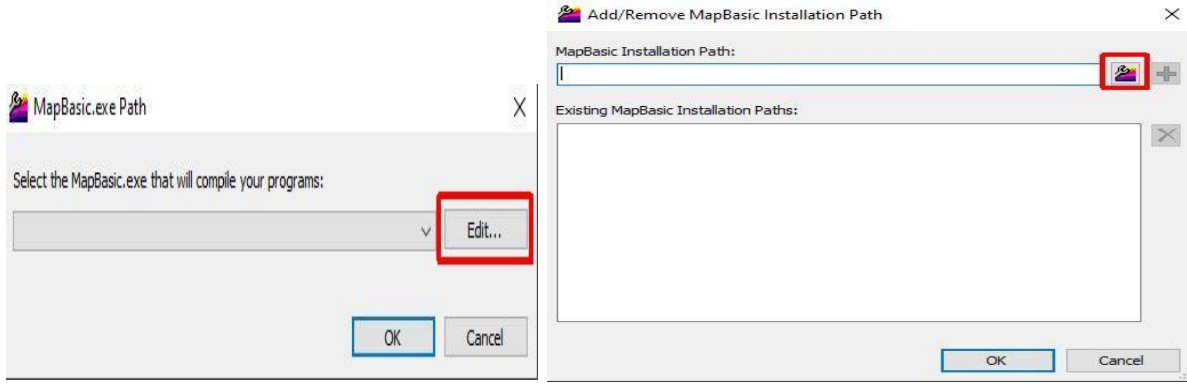
<https://mustafaozctin.wordpress.com/mapbasic/mapbasicide-en/>

Program kurulduktan sonra ilk açıldığında bir hata penceresi çıkacaktır (Şekil 43). Hata mesajında MapBasic programının çalıştırılabilir (Execute) program dizinini bulamadığını, dizini belirtmeni istemektedir.



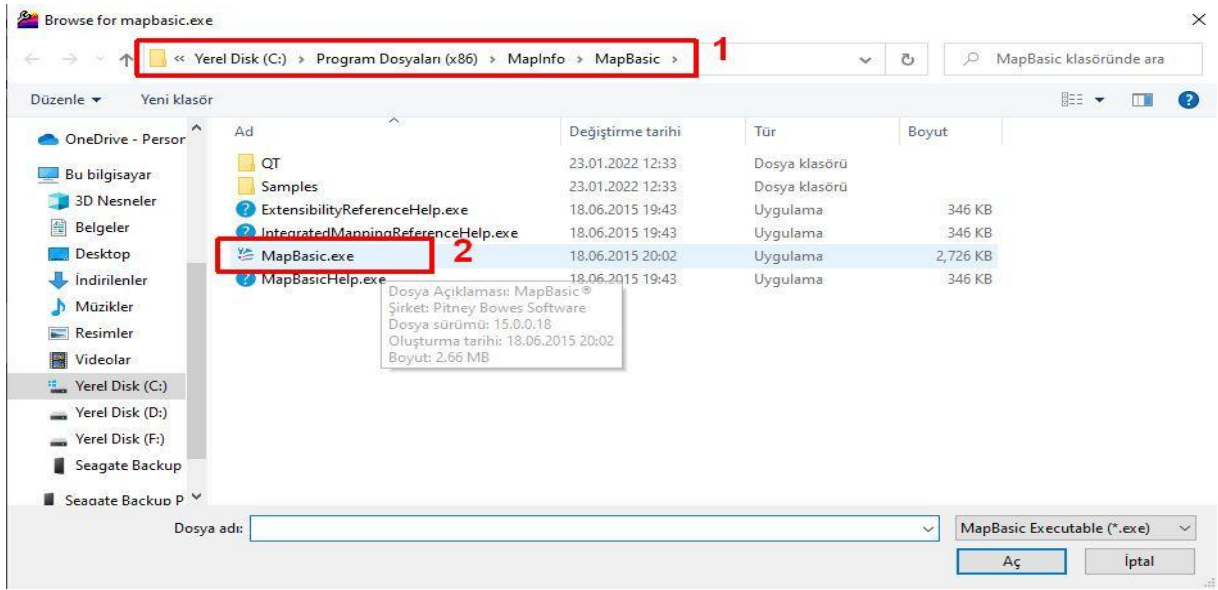
Şekil 43

Tamam düğmesine basıldığında, Şekil 44 sol resimdeki pencere çıkacaktır. Kırmızı ile çerçevelenmiş olan *Edit* düğmesine basıldığında Şekil 44 sağ resimdeki pencere açılacaktır. Şekil 44 sağ resimdeki kırmızı çerçevelenmiş (İngiliz anahtarı sembolü olan düğme) düğmeye basılmalıdır. Düğmeye basıldığında Şekil 49'deki pencere açılacaktır.

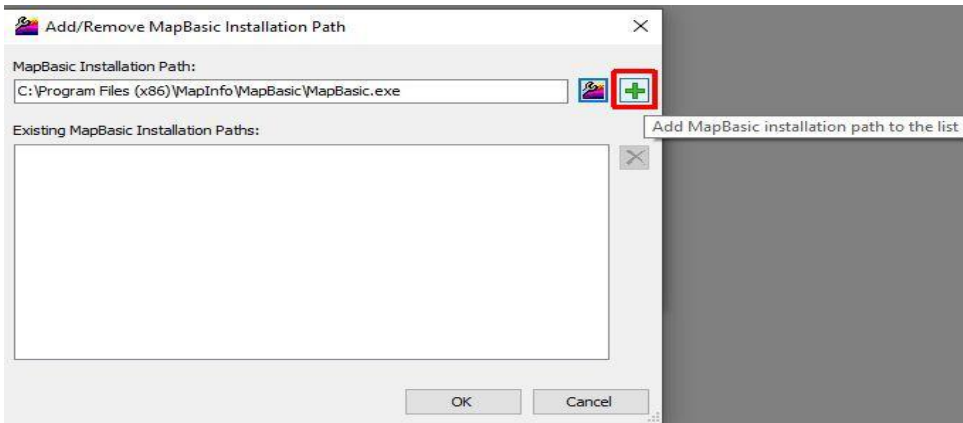


Şekil 44

Şekil 45 1 numaralı dizine gidilmeli ve 2 numaralı Exe uzantılı MapBasic dosyası seçilip Aç tuşuna basılmalıdır.



Şekil 45

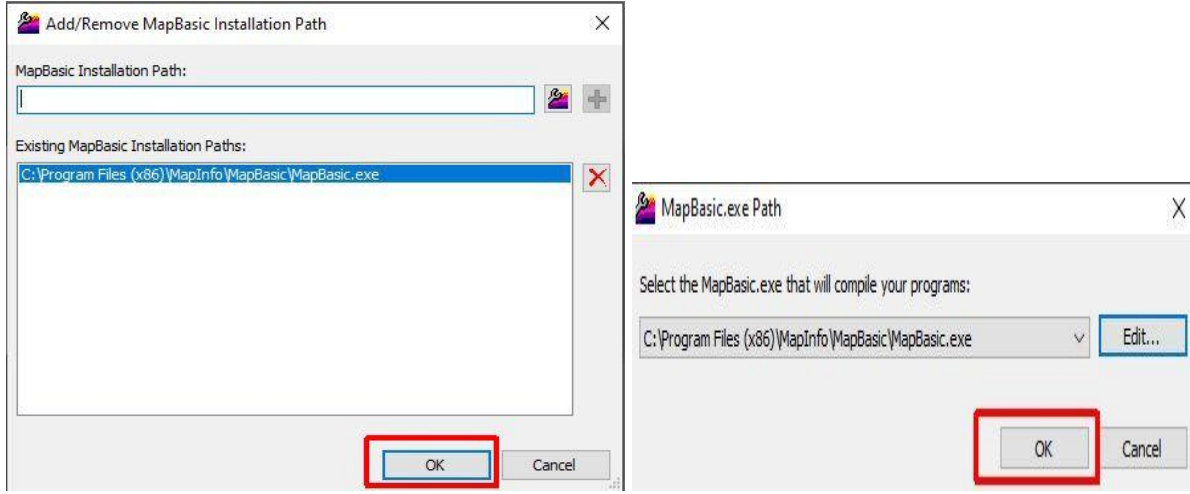


Şekil 45 aç tuşuna basıldığında Şekil 46'deki pencere açılacak. Pencerde kırmızı çerçeve ile belirtilen yeşil artı tuşuna basıldığında

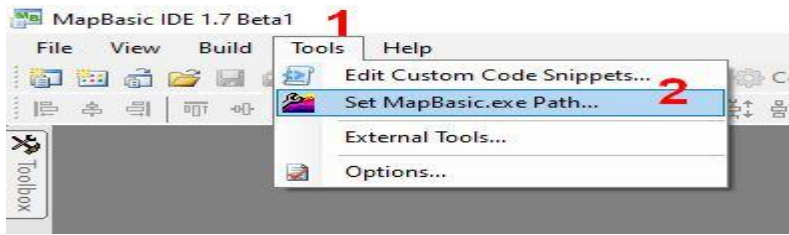
Şekil 46

Şekil 47 penceresinde görüldüğü gibi derleme için kullanılacak MapBasic.Exe çalıştırılabilir dosya dizini programa eklenmiş olacaktır. Pencerde Ok düğmesine ve yeni

açılan (Şekil 47 sağ resim) penceredeki Ok düğmesine basıldığında artık, derleyicinin adresi programa tanıtılmış olacak.



Şekil 47

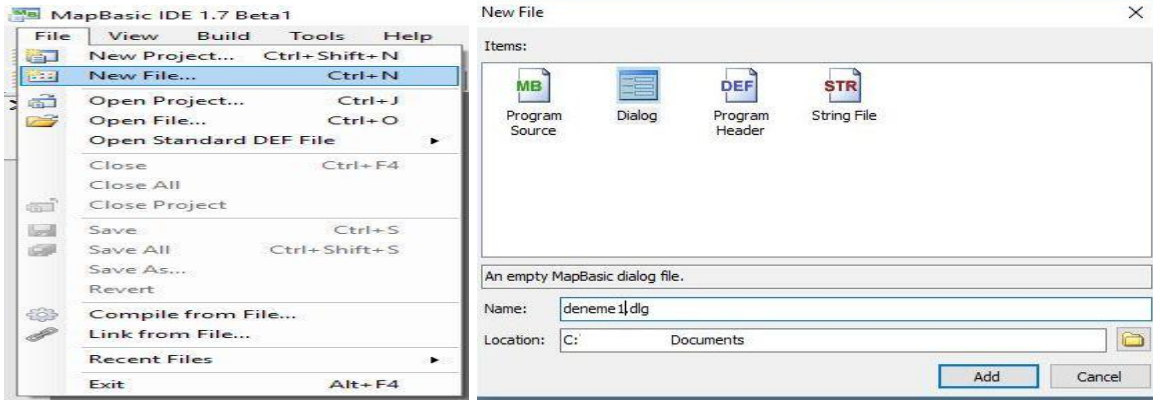


Şekil 48

Eğer Şekil 43'deki pencereyi kapatıp işleme devam ettiyseniz, Şekil 48'da 1 ve 2 numaralı menüleri kullanıp derleyici dosya ayarlaması

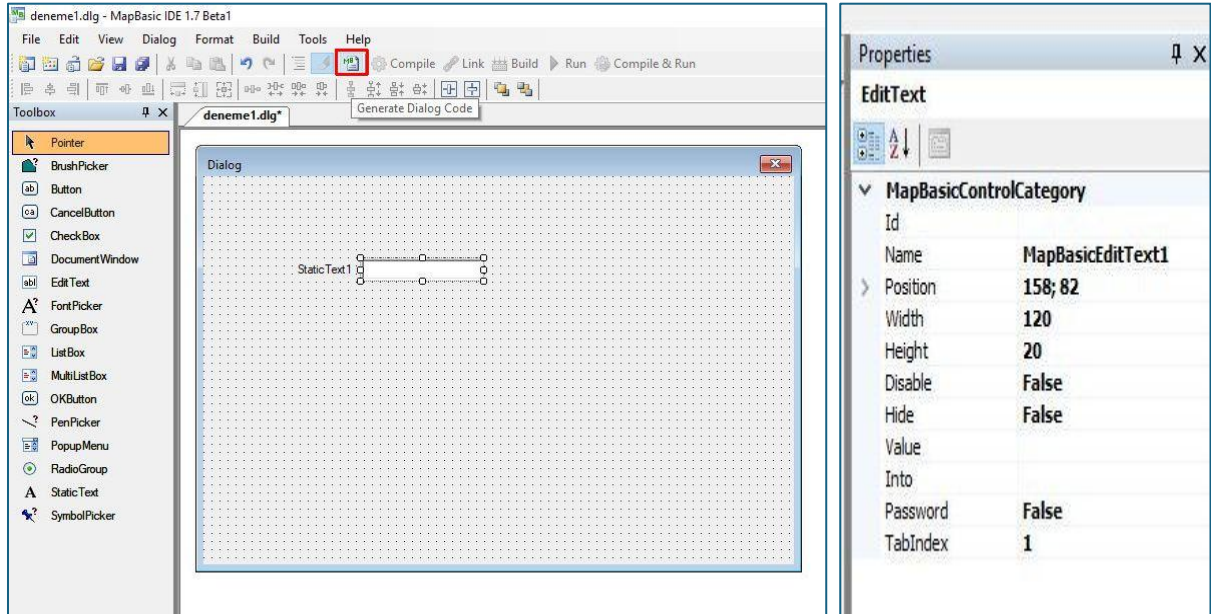
yapabilirsiniz.

MapBasic IDE programı çalıştırıldığında açılan ara yüzde *File* menüsü alt menülerinden *New File* seçildiğinde Şekil 49sağdaki resimde görülen *New File* penceresi açılacaktır. *New File* penceresi MapBasic IDE kullanılarak yapılabilecek olan işlemleri listeler. Dialog kontrolleri kullanılarak form tasarımı yapabilmek için *Dialog* seçilir, oluşturulacak dialog için bir isim ve kaydolacağı dizin belirtilir (Şekil 49 sağ resimde deneme1 ismi verildiği görülüyor).



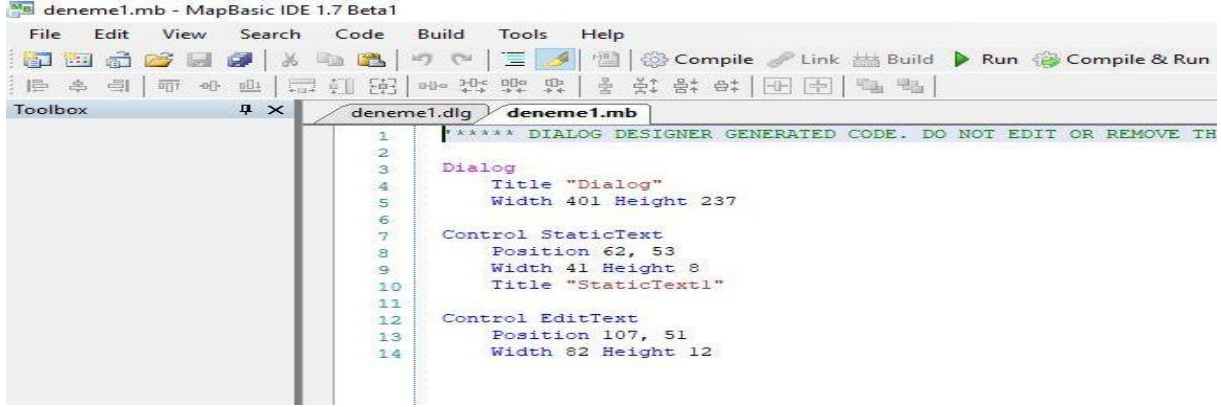
Şekil 49

Şekil 50 form tasarımı için ekrana gelen pencereyi göstermektedir. *Toolbox* araç kutusunda tüm Dialog kontrollerinin listelenmiş olduğunu görülmektedir. *Toolbox* içinden seçilen araç ekranın sağında kalan Dialog başlıklı form üzerine sürüklenip bırakılması aracın form üzerinde yerleşmesi için yeterli olacaktır. Eklenen aracın boyutları ekranın sağında ki *Properties* (özellikler) penceresinden düzenlenebilir (Şekil 50 sağ resim).



Şekil 50

Tasarlanan forma dair MapBasic kodunu görmek için Şekil 50'de kırmızı kutu içine alınan *Generate Dialog Code* (Dialog kodu üret) düğmesine tıklanır. DLG uzantılı olan formu oluşturduğumuz dosyayla aynı isimli MB uzantılı mapbasic kod dosyası oluşur (Şekil 51).



Şekil 51

MapBasic IDE yazılımı üzerinde oluşturulan ara yüz kullanılarak program geliştirilebilir. Fakat MapBasic IDE yazılımı kurulduğu bilgisayarın özelliklerine göre farklı hatalar oluşturmaktadır. MapBasic IDE yazılımını form tasarlayıp formun MapBasic kodunu üretmesi için kullanılması, üretilen formun MapBasic kodunun kopyalanıp MapBasic.exe programı içine yapıştırılması ve gereken program kodunun MapBasic.exe içinde yazılması daha hatasız işlemlerin oluşmasını sağlayacaktır.

Formda Event (Olay) Kontrolü

Form, hazırlanan programın kullanıcı ara yüzüdür. Kullanıcı form kullanarak veri girişi, veri seçimi, onay verme, düğmeye basma gibi form araçlarının olay özelliklerine bağlı olarak işlemler yapar. Form tasarımının yapılmasında, formun kullanıcı tarafından daha etkin ve anlaşılır olması düşünülmelidir. Kullanıcının form üzerinde yaptığı işlemlerin kontrolü ve sürdürülmesi olay kontrolü ile sağlanır. Örneğin kullanıcının form üzerinde girdiği verinin kayıt altına alınması için düğmeye basması bir olaydır. Bu olay sayesinde program kodu içinde verilerin işlenmesi sağlanacaktır.

Tasarlanan formlarda olayları yönetmek için daha önceden belirlenmiş bazı tanımlar, sabitler ve fonksiyonların çağrılabilmesi/kullanılabilmesi için Defination File (tanım dosyaları) gerekir. Bu dosyalar Mapbasic programı için tanımlanmış dosyalardır. Bu dosyalar program satırları içinde en üstte yazılmalı ve program çalıştırıldığında tanım dosyasının çağrılması sağlanır. Bu dosyalardaki tanımlı sabit değişkenler, fonksiyonlar kendi isimleriyle kullanabilmek veya çağırabilmek için programın en başına tanımlanmalıdır. *Include* komutu ile tanım dosyasının ismi eklenir. En çok kullanılacak olan tanım dosyası *Mapbasic.def* dosyasıdır. Bu tanım dosyası Mapbasic dosyasının kurulu olduğu dizin içinde bulunan (Şekil 52) Defination (tanım) bilgilerini içerir (Şekil 53). Programın başına

Include "mapbasic.def"

veya

include "Mapbasic.def"

Komut satırı olarak eklendiği takdirde bu dosya içindeki sabitler ve fonksiyonlar kendi isimleriyle çağrılabilir. Include ifadesinin burada kullanılan Türkçe karşılığı manası *dahil et* anlamındadır.

Ad	Değiştirme tarihi	Tür	Boyut
QT	23.01.2022 12:33	Dosya klasörü	
Samples	23.01.2022 12:33	Dosya klasörü	
Enums.def	18.06.2015 19:44	MapBasic Include ...	12 KB
ERRORS.DOC	18.06.2015 19:44	Microsoft Word 9...	74 KB
ExtensibilityReference.qch	18.06.2015 19:43	QCH Dosyası	2,059 KB
ExtensibilityReference.qhc	18.06.2015 19:43	QHC Dosyası	303 KB
ExtensibilityReferenceHelp.exe	18.06.2015 19:43	Uygulama	346 KB
HTTPLib.def	18.06.2015 19:44	MapBasic Include ...	16 KB
HTTPTypes.def	18.06.2015 19:44	MapBasic Include ...	1 KB
ICONS.DEF	18.06.2015 19:44	MapBasic Include ...	25 KB
IMapInfoPro.def	18.06.2015 19:44	MapBasic Include ...	451 KB
IntegratedMappingReference.qch	18.06.2015 19:43	QCH Dosyası	2,845 KB
IntegratedMappingReference.qhc	18.06.2015 19:43	QHC Dosyası	303 KB
IntegratedMappingReferenceHelp.exe	18.06.2015 19:43	Uygulama	346 KB
MAPBASIC.BAS	18.06.2015 19:44	BAS Dosyası	113 KB
MAPBASIC.DEF	18.06.2015 19:44	MapBasic Include ...	77 KB

Şekil 52

```

MAPBASIC.DEF
100 Define SECONDS_PER_DAY 86400
101
102 -----
103 ' Colors
104 ' -----
105 Define BLACK 0
106 Define WHITE 16777215
107 Define RED 16711680
108 Define GREEN 65280
109 Define BLUE 255
110 Define CYAN 65535
111 Define MAGENTA 16711935
112 Define YELLOW 16776960
113
114 ' -----
115 'Maximum length for character string
116 ' -----
117 Define MAX_STRING_LENGTH 32767
118
119 ' =====
120 ' BrowserInfo() defines
121 ' =====
122 Define BROWSER_INFO_NROWS 1
123 Define BROWSER_INFO_NCOLS 2
124 Define BROWSER_INFO_CURRENT_ROW 3
125 Define BROWSER_INFO_CURRENT_COLUMN 4
126 Define BROWSER_INFO_CURRENT_CELL_VALUE 5

```

Şekil 53

Girilen Değerlerin Yazdırılması

```

dialogadnumaragirişivenote.MB
include "mapbasic.def"
dim numara as integer
dim ad as string
Dialog
  title "Veri Girişi"
  control statictext
    position 10,10
    title "Öğrenci No:"
  control edittext
    position 50,10
    value "No Gir"
    into numara
  control statictext
    position 10,30
    title "Ad :"
  control edittext
    position 50,30
    value "Ad Gir"
    into ad
  control okbutton
  control cancelbutton

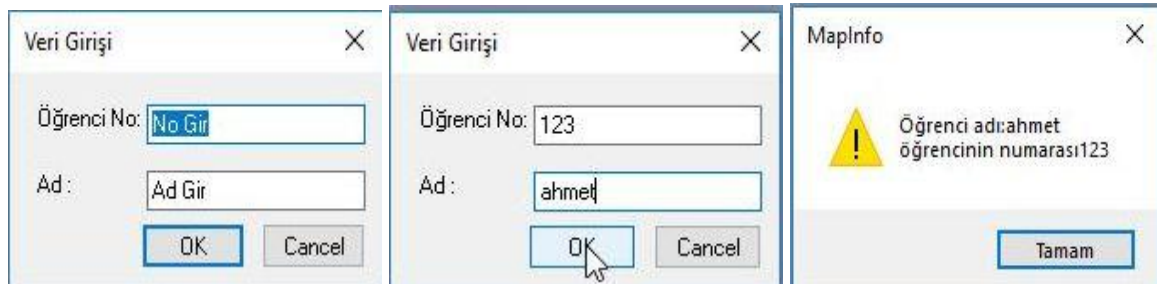
if commandinfo(CMD_INFO_DLG_OK) then
  note "Öğrenci adı:"+ ad+ Chr$(10)+
    "öğrencinin numarası"+numara
end if

```

Şekil 54

Şekil 54 bir form tasarımı örneğidir. Örneğin form çıktısı Şekil 55 en sol resimdir. İki adet *statictext* (Öğrenci No: ve Ad: yazıları), iki adet *edittext* (No Gir ve Ad Gir ifadelerinin olduğu veri girilebilen araçlar), *okbutton* ve *cancelbutton* formu oluşturan elemanlardır. *Edittext* özellikleri incelendiğinde *into* komutu görülecektir. *Into* komutlarından sonra yazılan değişkenler kodun en üst kısmında tanımlanmıştır. **Into sayesinde edittext içine girilen değerler değişkenlere aktarılmış oluyor.** Kodun ilk satırında Mapbasic.def tanım dosyası include komutuyla koda dahil edilmiş. Bu sayede kod içinde ki if – then bloğunda ki *sınama ifadesi* *commandinfo()* fonksiyonu içindeki ifadeyle kullanılabilir. If – then bloğunda kullanıcının *OkButton*

kontrolüne tıklamasının kontrolü yapılmaktadır Şekil 55 işlem sürecini göstermektedir.



Şekil 55

Şekil 54 kod içindeki olay, kullanıcının *OkButton* kontrolüne tıklamasıdır.

Button Kullanımı

Şekil 56, Şekil 54 örneğinde olduğu gibi kullanıcıdan veri girişi alınmaktadır. Şekil 54 örneğinden farklı olarak yordamlar kullanılmıştır. Program kodunun başında yordam

bildirimleri yapılmıştır. Bildirim için *declare* komutu kullanılmaktadır. Main yordamı, ana yordamdır. Mapbasic programı için sabit bir yordam adıdır. Eğer main yordam bildirimi yapılmışsa, Program kodu çalıştırıldığında ilk olarak *main* yordam bloğundan çalışmaya başlar. *Main* yordam bloğu ilk çalışacak kod bloğu olduğu için, form kodlarının *main* yordamı içinde tanımlanması daha anlamlıdır.

Şekil 54 program kodu içinde yordam tanımlanmamış ve değişkenler genel (statik) olarak tanımlanmıştır. Bu sebeple *into* komutu yardımıyla genel tanımlı değişkenlere, kullanıcının girdiği değerler atanmıştır.

Şekil 56 program kodu içinde main yordam bloğunda form kod satırları yazılmıştır. Bu sayede program ilk çalıştığında main yordamı içindeki form kodları çalışacaktır. Form kontrolleri içine kullanıcının girdiği verilerin değişkenlere aktarılması işlemi *goster* yordamı içinde yapılmıştır.

Şekil 56 **program kodu içinde olay** *Button* kontrolüne tıklanmasıyla oluşacaktır. Şekil 56 program kod bloğunda 11 ile 15 satırları arasında *Button* kontrol tanımı vardır. 15. satırda *Button* kontrolüne tıklandığında *goster* isimli yordam çağırılmaktadır. *Calling* komutu sayesinde program 37. satırda başlayan *goster* yordamına gidecektir.

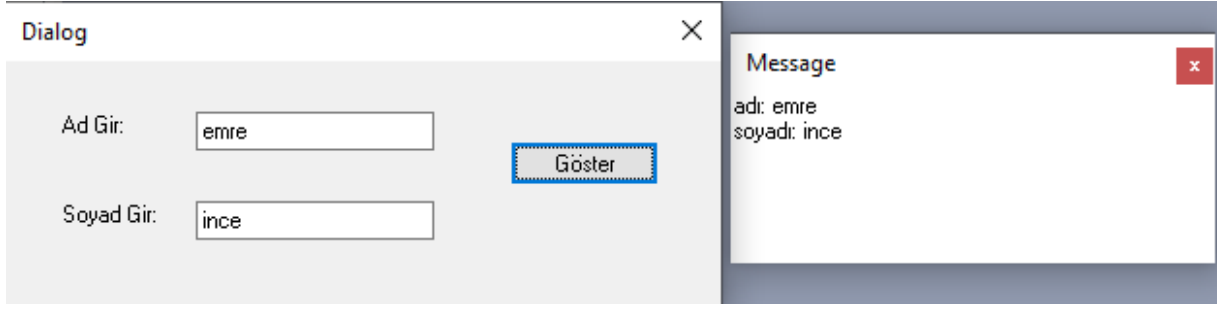
Şekil 56 21. ve 24 satırlarda kullanıcının ad bilgisini gireceği *Edittext* ve 31. ve 34. satırlarda kullanıcının soyadı bilgisinin girileceği *Edittext* form kontrollerinin tanımlanması vardır. Tanımlama kodlarına dikkat edilirse, her birisinde *Id* komutu kullanılmış. *Id* komutu identification (kimlik) kelimesinin kısaltılması olarak kullanılmaktadır. Her bir *Edittext* kontrolüne *id* komutu ile kimlik numarası verilmesi sağlanıyor. Kimlik numarası sayesinde, kullanıcının form kontrolüne e girdiği verinin okunması, kontrole veri aktarılması işlemleri yapılabilir. Şekil 56 40. ve 41. satırda *readcontrolvalue()* fonksiyonu ile *Edittext* e girilen verinin okunması sağlanmış olur. *Readcontrolvalue()* fonksiyonu içine form kontrolünün *id* numarası girilir. *Id* numarası ile kullanıcının girdiği değer okunması ve program kodunda olduğu gibi bir değişkene aktarılması sağlanır.



Button form elemanının *OkButton* elemanından farkı, *Button* elemanına basıldığında form kapanmaz. Fakat *Ok Button* elemanına basıldığında form kapanır.

```
4 include "mapbasic.def"
5 declare sub main
6 declare sub goster
7 sub main()
8     Dialog
9     Title "Dialog"
10    Width 246 Height 77
11    Control Button
12        Title "Göster"
13        Position 173, 25
14        Width 51 Height 14
15        calling goster
16    Control StaticText
17        Position 19, 16
18        Width 27 Height 8
19        Title "Ad Gir:"
20
21    Control EditText
22        Id 1
23        Position 65, 16
24        Width 82 Height 12
25
26    Control StaticText
27        Position 19, 44
28        Width 38 Height 8
29        Title "Soyad Gir:"
30
31    Control EditText
32        Id 2
33        Position 65, 44
34        Width 82 Height 12
35 end sub
36
37 sub goster
38     dim adtut,soytut as string
39     if readcontrolvalue(1)<>" " and readcontrolvalue(2) then
40         adtut=readcontrolvalue(1)
41         soytut=readcontrolvalue(2)
42         print chr$(12)
43         print "adı: " +adtut+chr$(10)+ "soyadı: " +soytut
44     end if
45 end sub
```

Şekil 56



Şekil 57

Kullanıcı tarafından girilen değerlerin kontrolü ve Yordam Kullanımı

Şekil 56 örneğinde kullanıcıdan öğrenci numarası ve ad bilgisinin girilmesi isteniyor. Girilen kod bilgisi incelendiğinde öğrencinin numarasının girişi kontrol edilmiyor. Numara yerine sözel bir ifade girilebilir ya da Edittext sahaları (metin kutuları) boş bırakılsa bile ekranda bir değer gösterimi olur. Değer girilirse boş bilgi olacak şekilde mesaj penceresi gözükabilir. Bunu önlemek için sına yapı kullanılabılır.



Şekil 56 örneğinde ve Şekil 58 örneğinde main yordamı kullanılmıştır. Program kodu içinde Main (ya da main) yordamı kullanıldığı bildirilmiş ise (declare – deklare edilmişse) o takdirde program ilk önce Main yordamından başlanır. Bu yüzden form (dialog) kullanımının olduğu programlarda, dialog main içine yazılmalı, ya da form farklı bir yordam içinde yazılıp main içinden çağırılmalıdır.


```

dialogadnumaragirisikontrol.MB x
1  include "mapbasic.def"
2  declare sub kontrol
3  declare sub main
4
5  'main yordamı program çalıştığı anda çalışan kod
6  sub main
7      Dialog
8          title "Veri Girişi"
9          control statictext
10         position 10,10
11         title "Öğrenci No:"
12         control edittext
13         position 50,10
14         value "No Gir"
15         id 1
16         control Statictext
17         position 10,30
18         title "Ad :"
19         control edittext
20         position 50,30
21         value "Ad Gir"
22         id 2
23         control Button
24         title "Ekranaya Yaz"
25         calling kontrol '****Yordamın çağrıldığı yer
26     end sub
27
28     sub kontrol
29         'Boş değer girilmediğinin ve kutu içindeki değerlerden farklı bir
30         'değer girdiğinin kontrolü
31         if readcontrolvalue(1)<>" " and readcontrolvalue(1)<>"No Gir" then
32
33             if readcontrolvalue(2)<>" " and readcontrolvalue(2)<>"Ad Gir" then
34
35                 Note "Öğrenci numarası: " + readcontrolvalue(1)+chr$(10)+"adı: " + readcontrolvalue(2)
36
37             else
38                 Note "Öğrenci adı girin!"
39             end if
40
41         else
42             Note "Öğrenci numarası girin!"
43         end if
44
45     end sub

```

Şekil 58

Şekil 58 örneği daha önceki (Şekil 57) kod bloğunun biraz geliştirilmiş halidir. Kod içeriğine yordam (procedure) eklenmiştir. *Main* isimli yordam genel bir yordamdır. Program ilk çalıştığında *main* yordamı içine girer ve uygulanacak işlem adımları *Main* yordamı içinde tasarlanır. *Main* yordamı içinde form tasarımı vardır. 2 adet *Edittext* (metin kutusu) *id*

özelliğine 1 ve 2 olacak şekilde kimlik numaraları verilmiştir. Bu *Edittext* değerlerindeki verilere ulaşmak için *id* değerleri kullanılacaktır. Kodda daha önce kullanılmamış bir form elemanı olan *Button* kullanılmış.



Button kontrolü bir düğme kontrolüdür. *Button* aynı *OkButton* kullanımında olduğu gibi bir işlemin/olayın oluşmasını sağlar. *Button* kontrol tanımında *Calling kontrol* ifadesi, olayın oluşmasını sağlamıştır (Şekil 58 25. satır). Eğer *Button* kontrolüne basılırsa, program kod satırı *kontrol* yordamına gidecektir. Bir nevi, *Button* kontrolüne basıldığında *kontrol* yordamı çağırılmış olacaktır.

Şekil 58 28. Satır ile 45. Satır arasında *kontrol* yordamı kod bloğu bulunmaktadır. Yordam için *Edittext* kontrolleri içine veri girilip girilmediğinin kontrolü yapılmakta ve eğer değerler girildiğiye öğrencinin numarası ve adı ekrana uyarı penceresiyle yazılmaktadır.

Yordam içinde *Edittext* kontrolleri içine veri girişinin yapılıp yapılmadığının kontrolün yapmak için, *Edittext* içindeki verinin okunması gerekmektedir. Veriyi okumak için *Readcontrolvalue()* komutu kullanılır. *Readcontrolvalue()* bir fonksiyondur. *Readcontcontrolvalue()* fonksiyonu içine aldığı kontrol id değeriyle kontroller içindeki değerlere ulaşabilir. Örnekte öğrenci numarasının girileceği *Edittext id* değeri 1 olarak belirlenmiştir (Şekil 58 15. satır). *Readcontrolvalue(1)* komutu ile öğrenci numarasının girildiği *Edittext* içindeki değere ulaşacaktır. Öğrenci adının girileceği *Edittext id* değeri 2 olarak belirlenmiştir (Şekil 58 22. satır). *Readcontrolvalue(2)* komutu ile öğrencin adının girildiği *Edittext* içindeki değere ulaşacaktır.

Edittext kontrollerinin tanımları yapılırken, *value* komutu kullanılmıştır. *Value* komutu, kontrolde gözükecek değer belirlenmesini sağlar. Bu sayede kullanıcının veri girmesi gereken yer ve girilecek veri bilgisi gösterilmiş olacaktır (Şekil 58 14. ve 21. satırlar).

Kod içinde iki adet *if – then* karşılaştırma bloğu bulunmaktadır. İlk blokta öğrenci numarasının girilip girilmediği, içteki *if – then* bloğunda ise öğrenci adının girilip girilmediğinin sınaması yapılmaktadır.

If readcontrolvalue(1) <> "" and readcontrolvalue(1) <> "No Gir" then

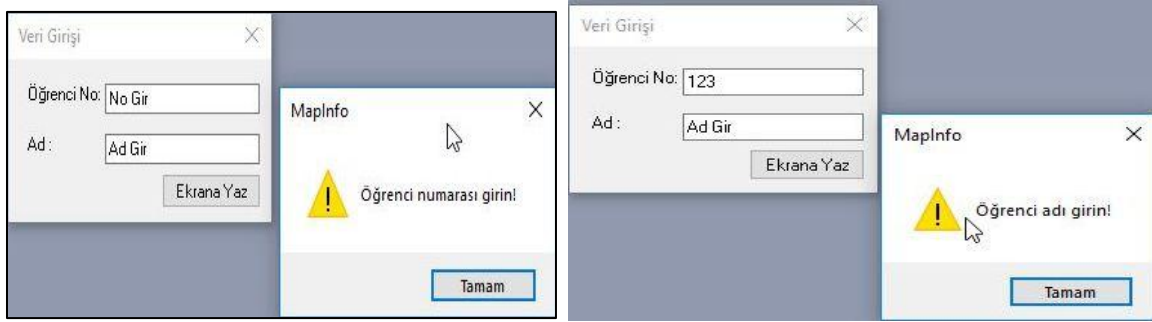
End If

Yukarıdaki *if then – end if* kod bloğu Şekil 58 31. satırda başlamaktadır. <> ifadesi “eşit değilse” anlamıdadır. *If – then* satırı okunursa:

I id numarasına sahip kontrol "" (boş) 'a eşit değilse VE I id numarasına sahip kontrol "No Gir" değerine eşit değilse aşağıdaki kod bloğunu çalıştır.

İçte kalan ikinci *If then – End If* kod bloğu ise 2 id değerine sahip *Edittext* kontrolüne veri girişinin kontrolü için yapılmıştır.

Şekil 58 41. Satırda, ilk *If* sınaması olmadığı durumda yapılacakları belirlemek için *else* komutu kullanılmıştır. *Else* komutu çalışması için: 1 id numarasına sahip kontrol “” (boş)’a eşit olması yeterli olacaktır. Ya da 1 id numarasına sahip kontrol “No Gir” değerine eşit olması yeterli olacaktır. Bu durumda .Şekil 59 sol resimde olduğu gibi kullanıcıya uyarı mesajı girilecektir. .Şekil 59 sağ resimde, iç if bloğunun kontrolü sonucu veri girilmemesi veya “Ad Gir” kalması durumunda kullanıcıya çıkacak uyarı mesajı gözükmektedir.



Şekil 59



Şekil 60

Program Kod İçinde Hata Yakalama İşlemleri

Programlama dillerinde yazılan kod esnasında hata oluşabilir. Hata oluşabileceğini düşünülen yerde hatanın yakalanıp programın kapanmayacağı şekilde bir uygulama geliştirilmelidir. Programlama dillerinde genelde Try – Catch bloğu olarak ifade edilen kod blokları sayesinde hata yakalanıp kullanıcı uyarılabilir. Şekil 58 kullanıcının metin kutularını boş bırakıp bırakmadığını kontrol ediyor. Kullanıcının formdaki bir metin kutusuna sayısal veri girmesi gerekirken alfabetik veri girmesi hatası kontrol edilmiyor.

Harita yapımı amaçlı uygulamalar için tasarlanmış formlarda kullanıcıdan sayısal veri girişi istenir. Eğer kullanıcı sözel veri giriyorsa bu durum belirlenmeli ve program kapanmadan kullanıcı uyarılıp işlemin tekrarı sağlanabilir.

Mapbasic programlama dili içinde bu tip sorunların çözümleri için *OnError Goto, Err()* ve *Error\$()* parametreleri kullanılır.

OnError Goto parametresinden sonra bir kelime kullanılır. Kullanılan kelime ile bir blok yapı oluşturulur. ifade 7 hata yakalama kullanımına örnektir. OnError Goto hatavar ile hatavar: arasında kalan program satırları içinde hata oluşursa program direkt olarak hatavar: satırının altında kalan satırdan devam eder.

ifade 7

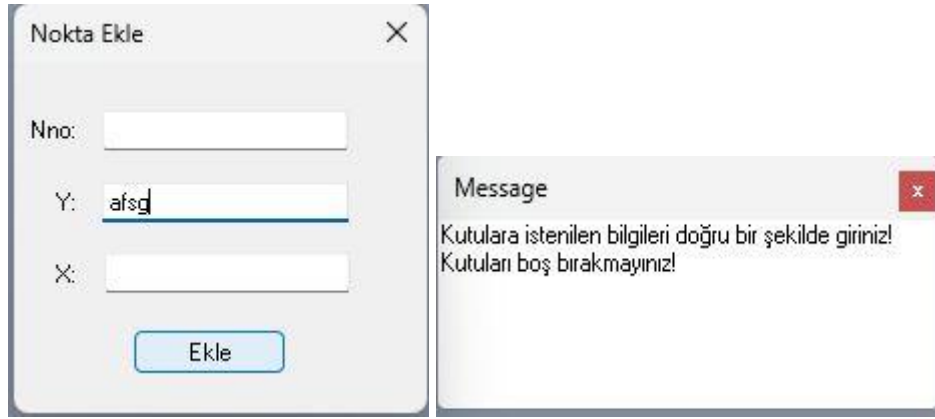
OnError Goto hatavar
{hatanın olabileceği
program kodları}
hatavar:

1	include "mapbasic.def"	34	Control EditText
2	declare sub main	35	Id 2
3	declare sub kontrol	36	Position 29, 36
4	sub main	37	Width 82 Height 12
5	Dialog	38	
6	Title "Nokta Ekle"	39	Control EditText
7	Width 137 Height 107	40	Id 3
8		41	Position 30, 58
9	Control EditText	42	Width 82 Height 12
10	Id 1	43	end sub 'main kapanışı
11	Position 29, 14	44	
12	Width 82 Height 12	45	sub kontrol
13		46	dim x_al,y_al as float
14	Control Button	47	dim nno_al as string
15	Title "Ekle"	48	onerror goto hatavar
16	Position 39, 81	49	nno_al=readcontrolvalue(1)
17	Width 51 Height 14	50	y_al=readcontrolvalue(2)
18	calling kontrol	51	x_al=readcontrolvalue(3)
19	Control StaticText	52	hatavar:
20	Position 5, 16	53	if err()=854 then
21	Width 20 Height 8	54	call temizle
22	Title "Nno:"	55	elseif err()=0 then
23		56	print "NNo:" + nno_al + "Y:" + y_al + "X:" + x_al
24	Control StaticText	57	end if
25	Position 14, 38	58	end sub
26	Width 12 Height 8	59	sub temizle
27	Title "Y:"	60	print chr\$(12)
28		61	print "Kutulara istenilen bilgileri doğru bir şekilde giriniz!"
29	Control StaticText	62	print "Kutuları boş bırakmayınız!"
30	Position 14, 60	63	alter control 1 value " "
31	Width 12 Height 8	64	alter control 2 value " "
32	Title "X:"	65	alter control 3 value " "
33		66	end sub

Şekil 61

Şekil 62 Nokta grafik objesinin haritaya eklenmesi için kullanıcıdan koordinat bilgilerinin alındığı form. Bu formda Y ve X koordinat değerlerinin girileceği kısımda alfabetik karakter girilip Ekle düğmesine tıkladığında metin kutuları silinip, kullanıcıya uyarı mesajı gönderiliyor. Bu işlemin yapılabilmesi için hata yakalama program satırları Şekil 61 48. ile 52. satırlar arasında kalıyor. Belirtilen kod satırları arasında hata çıkabilir. *y_al* ve *x_al* isimli değişkenler float (reel sayı) veri tipinde. Eğer kullanıcı alfabetik değer girer ve ekle tuşuna basarsa hata oluşur. Karakter tipindeki veri reel sayı tipindeki değişkene aktarılmaz.

Hata yakalandığında program 53. Satıra gidiyor. *Err()* fonksiyonu hatanın sayısal kod numarasını belirtiyor. 854 hatası, değişkene farklı veri tipinde değer atanması sonucu yazılımın verdiği hata kodudur. Eğer bir hata yoksa *Err()* 0 değeri döndürüyor.



Şekil 62

Girilen Rasyonel Sayı Halindeki Derece Açı Birimini Derece – Dakika – Saniye Haline Dönüştürülmesi ve Yordam Kullanımı

Şekil 63 programı kod örneğinde, kullanıcının gireceği derece açı biriminde reel sayının (ondalıklı sayı), derece – dakika – saniye formatında çıktısının alınması sağlanmıştır. İşlem için iki adet yordam kullanılmıştır. Yordamlardan bir tanesi main (ana) yordamı bir diğeri de dönüşüm yordamı. Main yordamı içinde, daha önceki örneklerde olduğu gibi, form tasarımı için kod bloğu konulmuştur.

Main yordamı içindeki form kod bloğu içinde kod içindeki *olay* tanımlanmıştır. Form kod bloğunda tanımlanan *Button* kontrolü tanımı içinde *calling* komutu ile *donusum* yordamı çağırılmaktadır. Bu sayede kullanıcının *Button* kontrolüne tıklanmasıyla *donusum* isimli yordam çağırılmış olacaktır.

donusum yordamı içinde 34. satır ile 43. satır arasında if – End if bloğu bulunmaktadır. If – End If bloğunda, kullanıcının veri girişi yapıp yapmadığının kontrolü yapılmaktadır. Bu kontrolü yapabilmek için 34. satırdaki:

if readcontrolvalue(1)<> “” and readcontrolvalue(2)<> “” then →komut satırı kullanılmıştır

- readcontrolvalue(1), id değeri 1 olan kontrolün içindeki bilginin okunmasını sağlar,
- readcontrolvalue(1)<> “”, id değeri 1 olan kontrolün içindeki bilginin boş değerden farklı olup olmadığını sorgulamak için kullanılır,
- readcontrolvalue(2), id değeri 2 olan kontrolün içindeki bilginin okunmasını sağlar,
- readcontrolvalue(2)<> “”, id değeri 2 olan kontrolün içindeki bilginin boş değerden farklı olup olmadığını sorgulamak için kullanılır.

Satır okunduğunda:

“id değeri 1 olan kontrolün içi boştan farklıysa ve id değeri 2 olan kontrolün içi boştan farklıysa”

anlamına gelmektedir. If-then sınama cümleciğinde, *and* mantıksal operatörü kullanılmıştır. And operatörüyle hem 1 hem de 2 id değerlerine sahip kontrollerin boş olmadığı takdirde ibaresi sağlanmıştır. Eğer sınama gerçekleşirse, 35 ile 40 numaralı satırlar arasındaki kod bloğu çalışacaktır.

And operatörü sayesinde, iki kontrolden biri dahi boş bırakılmış olursa, *else* ile *End If* komutları arasındaki kod satırları çalışacaktır. Örnek kodda 41 ile 43 numaralı satırlar arasındaki kod satırları çalışacak.



donusum yordamı içinde *id* değeri 3 olan *Edittext* içeresine veri aktarımının yapılması için, Şekil 63 40. Satırdaki *Alter* komutu kullanılmıştır. *Alter* komutu, var olan kontrolün bir özelliğini değiştirmek için kullanılır.

```

dialogderece_dakika_saniye.MB x
1  include "mapbasic.def"
2  declare sub main
3  declare sub donusum
4
5  sub main
6      Dialog
7          title "Açı --> Derece - Dakika - Saniye"
8      control statictext
9          position 10,10
10         title "Açı Gir"
11     control edittext
12         position 40,8
13         value "açı Gir"
14         id 1
15     control button
16         position 55,28
17         title "Dönüştür"
18         id 2
19         calling donusum
20     control statictext
21         position 10,50
22         title "D-D-S"
23     control edittext
24         position 40,48
25         id 3
26     control cancelbutton
27         title "Çıkış"
28 end sub
29
30 sub donusum
31     dim deger,saniye as float
32     dim derece, dakika as integer
33     dim sonuc as string
34     if readcontrolvalue(1) <> "" and readcontrolvalue(1) <> "açı gir" then
35         deger=readcontrolvalue(1)
36         derece=fix(deger)
37         dakika=fix((deger-derece)*60)
38         saniye=((deger-derece)*60-dakika)*60
39         sonuc= derece+ chr$(176) + " " + dakika +chr$(39)+ " " + saniye +chr$(34)
40         alter control 3 value sonuc
41     else
42         note "Bir açı değeri girmelisiniz!"
43     end if
44
45 end sub
46

```

Şekil 63



Şekil 64

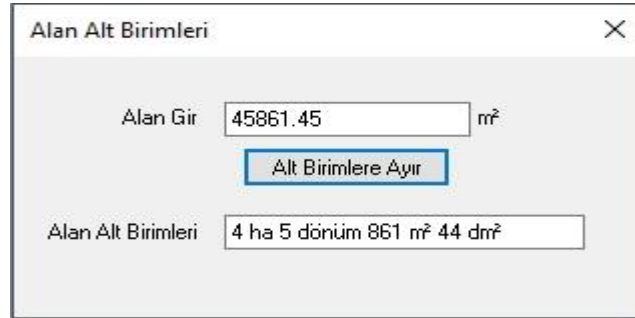
Kullanıcıdan Alınan Reel Sayı Veri Tipindeki Alan Değerinin Alt Birimlere Ayrılması

Kullanıcı 1 numaralı kontrol olan edittext kontrolüne girdiği reel sayı (ondalıklı sayı) metre kare alan birimindedir. Metre kare biriminde verilen alan değerinin alt birimlere dönüştürülmesi için metre kare birimi ile alt birimler arasındaki katsayıların bilinmesi gerekir.

$$1 \text{ hektar} = 10000 \text{ m}^2$$

$$1 \text{ dönüm} = 1000 \text{ m}^2$$

$$1 \text{ desimetrekare} = 0.01\text{m}^2$$



Şekil 65


```

1 include "mapbasic.def"
2 declare sub main
3 declare sub donustur
4 sub main
5     Dialog
6         Title "Alan Alt Birimleri"
7         Width 210 Height 92
8
9     Control StaticText
10        Position 36, 17
11        Width 30 Height 8
12        Title "Alan Gir"
13
14    Control EditText
15        Id 1
16        Position 70, 16
17        Width 82 Height 12
18
19    Control StaticText
20        Position 11, 58
21        Width 55 Height 8
22        Title "Alan Alt Birimleri"
23
24    Control EditText
25        Id 2
26        Position 70, 56
27        Width 120 Height 12
28    Control StaticText
29        Id 3
30        Position 154, 17
31        Width 52 Height 8
32        Title "m"+chr$(178)
33    Control Button
34        Title "Alt Birimlere Ayr"
35        Position 76, 32
36        Width 69 Height 14
37        calling donustur
38    end sub
39 sub donustur
40    dim hektar,donum,m2,dm2,deger as float
41    dim sonuc as string
42    deger=readcontrolvalue(1)
43    hektar=fix(deger/10000)
44    donum=fix((deger/10000-hektar)*10)
45    m2=fix(deger - hektar*10000-donum*1000)
46    dm2=fix((deger-fix(deger))*100)
47    sonuc=hektar + " ha " + donum + " dönüm " + m2 + " m" + chr$(178) + " " + dm2 + " dm" + chr$(178)
48    alter control 2 value sonuc
49 end sub

```

Şekil 66

Kullanıcıdan Alınan Veri kullanılarak Hesaplama

Şekil 67 ve Şekil 68 bir program kodunun iki resme ayrılmış halidir. Program kodu sayesinde kullanıcıdan alınacak iki noktanın X ve Y koordinatları kullanılarak semt açısının hesaplanması sağlanmaktadır.

Programda bir tanesi main (ana) olmak üzere iki ayrı yordam ve bir adet fonksiyon kullanılmaktadır. Main yordamında form dialog tasarım kodu bulunmaktadır. Bu sayede, program ilk çalıştığında main yordamı içine girmiş olacak. Main yordamındaki kod çalışarak, form kullanıcının karşısına çıkacak.

Program içindeki **olay Button** kontrolüne basılmasıyla oluşacak. Kullanıcı form üzerindeki *Button* kontrolüne bastığında, 11 ve 15 numaralar arasındaki kod çalışacak. *Button* kontrolüne basıldığında 15. Satırda ki *calling kontrol* komut satırı çalışacak ve kod 66. Satırdaki *kontrol* yordamına aktarılacak.

kontrol yordamı içinde kullanıcının her iki nokta için koordinat gireceği *Edittext* kontrollerine değer girilip girilmediğinin kontrolleri yapılmaktadır. Kontrollerin yapılması için IF – End If kod blokları kullanılmaktadır. Eğer dört kontrole de değer girilirse, 71 ile 76 satır numaraları arasındaki kod bloğu çalışacaktır. Eğer kullanıcı herhangi bir *Edittext* içine veri girmemiş ise o takdirde 77 ile 82 numaralı kod satırları arasındaki kodlar ile kullanıcı uyarılacaktır.

kontrol yordamı içinde kullanıcı 75 numaralı satırda *semt()* fonksiyonu çağırılmıştır. ***Yordam çağırılması için call komutu kullanılıyorken fonksiyon çağırılmak için fonksiyonun adının yazılması yeterli olacaktır.*** Fonksiyon geriye değer döndürür. 75 numaralı satırda *semt()* fonksiyonunun geriye döndürdüğü değer *semtsonuc* adlı float veri tipinde değişkendir. *semt()* fonksiyonunun tanımında:

```
function semt(Byval x1,y1,x2,y2 as float) as float
```

```
end function
```

İtalik ve kırmızı yazılmış olan ***as float*** ifadesi, fonksiyonun geriye döndüreceği veri tipini belirtiyor. Bu sebepten ötürü *semtsonuc* değişkeni float veri tipindedir.

```
1 include "mapbasic.def"
2 declare sub main
3 declare sub kontrol
4 declare function semt (byval x1,y1,x2,y2 as float) as float
5
6 sub main
7     Dialog
8     Title "Semt Hesabı"
9     Width 253 Height 128
10
11 Control Button
12     Title "Hesapla"
13     Position 107, 70
14     Width 51 Height 14
15     calling kontrol
16 Control StaticText
17     Position 9, 21
18     Width 46 Height 8
19     Title "Birinci Nokta"
20
21 Control StaticText
22     Position 9, 42
23     Width 44 Height 8
24     Title "İkinci Nokta"
25
26 Control EditText
27     Id 1
28     Position 56, 21
29     Width 82 Height 12
30
31 Control EditText
32     Id 3
33     Position 56, 42
34     Width 82 Height 12
35
36 Control EditText
37     Id 2
38     Position 150, 21
39     Width 82 Height 12
40
41 Control EditText
42     Id 4
43     Position 150, 42
44     Width 82 Height 12
45
46 Control StaticText
47     Position 77, 11
48     Width 33 Height 8
49     Title "Y Değeri"
50
```

Şekil 67

```

51 Control StaticText
52     Position 169, 11
53     Width 33 Height 8
54     Title "X Değeri"
55
56 Control StaticText
57     Position 75, 100
58     Width 26 Height 8
59     Title "Sonuç"
60
61 Control EditText
62     Id 5
63     Position 120, 98
64     Width 82 Height 12
65 end sub
66 sub kontrol
67     dim x1,y1,x2,y2 as float
68     dim semtsonuc as float
69     if readcontrolvalue(1)<>" " and readcontrolvalue(2)<>" " then
70         if readcontrolvalue(3)<>" " and readcontrolvalue(4)<>" " then
71             y1=readcontrolvalue(1)
72             x1=readcontrolvalue(2)
73             y2=readcontrolvalue(3)
74             x2=readcontrolvalue(4)
75             semtsonuc=semt(x1,y1,x2,y2)
76             alter control 5 value str$(format$(semtsonuc,"#.###"))
77         else
78             note "İkinci noktanın koordinatlarını girmelisiniz!"
79         end if
80     else
81         note "Birinci noktanın koordinatlarını girmelisiniz!"
82     end if
83 end sub
84 function semt(byval x1,y1,x2,y2 as float)as float
85     dim pi as float
86     pi=3.14159265358979
87     if (y2-y1)>0 and (x2-x1)>0 then
88         semt=atn((y2-y1)/(x2-x1))*200/pi
89     elseif (y2-y1)>0 and (x2-x1)<0 then
90         semt=200-atn((y2-y1)/(x2-x1)*(-1))*200/pi
91     elseif (y2-y1)<0 and (x2-x1)<0 then
92         semt=200+atn((y2-y1)/(x2-x1))*200/pi
93     else
94         semt=400-atn((y2-y1)/(x2-x1)*(-1))*200/pi
95     end if
96 end function

```

Şekil 68

	Y Değeri	X Değeri
Birinci Nokta	512456.921	4312648.091
İkinci Nokta	512591.036	4312501.334

Hesapla

Sonuç: 152.8635

Kullanıcıdan Alınan Veriler ile Hesaplama (Hem Yordam Hem de Fonksiyon Kullanımı)

Şekil 69 kullanıcının sol ve sağ resim, aynı kod dizinin ikiye bölünmüş halidir. Kod bloğu incelendiğinde üç adet yordam ve bir adet fonksiyon kullanılmıştır. Yordamlardan bir tanesi main yordamıdır. Main yordamı içinde form tasarımı yapılmıştır.

Nno	Y	X	Semt
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
			Uzunluk <input type="text"/>

Hesapla

Şekil 69

Nno sütunu kullanıcının gireceği nokta numarası sahaları, Y sütunu noktaya ait Y koordinatlarının girileceği sütun, X sütunu noktaya ait X koordinatının girileceği sütundur. Hesapla isimli düğmeye basıldığında *hesapla* yordamı çağrılacak (Şekil 71 satır 51). Şekil 71 80. satırda başlayan *hesapla* yordamı içinde sırasıyla ilk önce kullanıcının girdiği değerler değişkenlere aktarılıyor sonrasında *yaz* yordamı ve *semtacisi* fonksiyonu çağrılıyor. *yaz* yordamıyla *semt* isimleri tanımlı *statictext* kontrollerine yazılıyor. *semtacisi* fonksiyonu, *semt* açılarını hesaplayıp çağırıldığı yere geriye değer döndürüyor. Şekil 70 girilen koordinat değerleri sonucu bulunan değerlerin ekran görüntüsüdür.

Dialog

Nno	Y	X		
5	512344.574	4388215.142	(5-8)	169.328
8	512378.218	4388150.799	(8-5)	369.328
			Uzunluk	72.608

Hesapla

Şekil 70

```

arazyuz.mb
1 include "mapbasic.def"
2 declare sub main
3 declare sub hesapla
4 declare sub yaz
5 declare function semtacisi(Byval x1,y1,x2,y2 as float) as float
6 sub main
7     Dialog
8         Title "Dialog"
9         Width 479 Height 93
10        Control StaticText
11            Position 19, 18
12            Width 18 Height 8
13            Title "Nno"
14        Control StaticText
15            Position 73, 18
16            Width 10 Height 8
17            Title "Y"
18        Control StaticText
19            Position 135, 18
20            Width 10 Height 8
21            Title "X"
22        Control EditText
23            Id 1
24            Position 19, 30
25            Width 18 Height 12
26        Control EditText
27            Id 2
28            Position 54, 29
29            Width 46 Height 12
30        Control EditText
31            Id 3
32            Position 113, 28
33            Width 49 Height 14
34        Control EditText
35            Id 4
36            Position 19, 55
37            Width 18 Height 13
38        Control EditText
39            Id 5
40            Position 53, 55
41            Width 47 Height 12
42        Control EditText
43            Id 6
44            Position 113, 55
45            Width 49 Height 12
46        Control Button
47            Title "Hesapla"
48            Id 7
49            Position 193, 37
50            Width 51 Height 17
51            calling hesapla
52        Control StaticText
53            Id 12
54            Position 352, 15
55            Width 21 Height 8
56            Title "Semt"
57        Control StaticText
58            Position 342, 56
59            Width 31 Height 8
60            Title "Uzunluk"

```

```

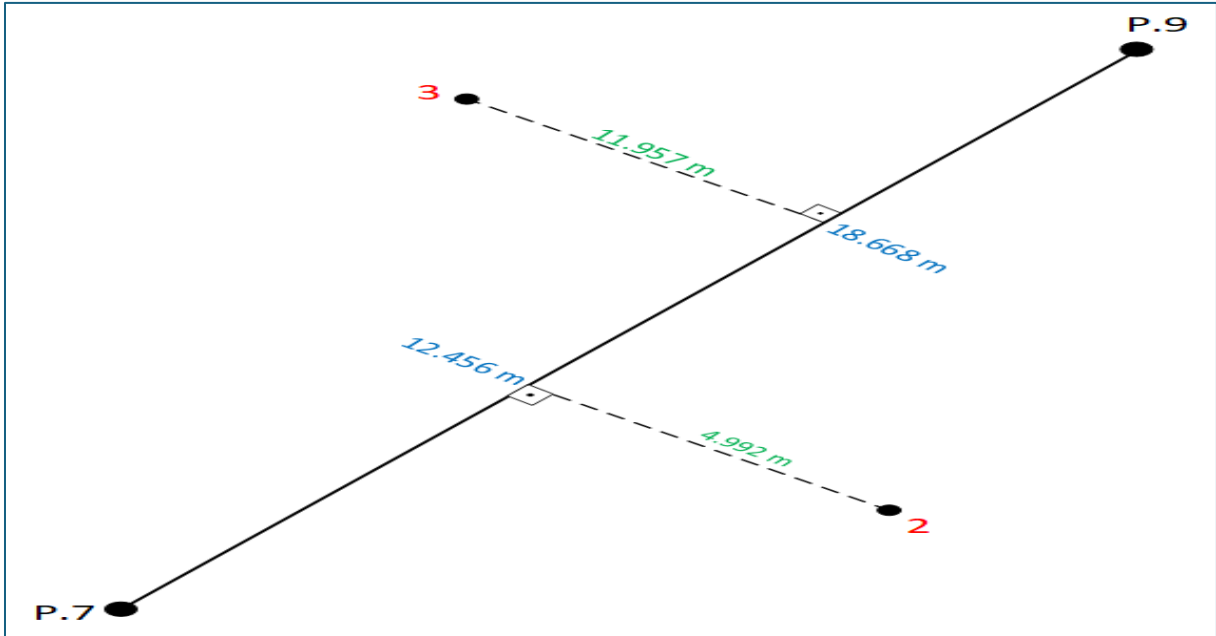
arazyuz.mb x
59      Width 31 Height 8
60      Title "Uzunluk"
61      Control EditText
62      Id 9
63      Position 386, 13
64      Width 73 Height 15
65      Control EditText
66      Id 11
67      Position 386, 55
68      Width 73 Height 15
69      Control StaticText
70      Id 13
71      Position 352, 33
72      Width 21 Height 8
73      Title "Semt"
74      Control EditText
75      Id 10
76      Position 386, 32
77      Width 73 Height 15
78  end sub
79
80  sub hesapla
81      dim nx1,ny1,nx2,ny2 as float
82      dim deg1,deg2 as float
83      nx1=readcontrolvalue(3)
84      ny1=readcontrolvalue(2)
85      nx2=readcontrolvalue(6)
86      ny2=readcontrolvalue(5)
87
88      call yaz
89
90      deg1=semtacisi(nx1,ny1,nx2,ny2)
91      deg2=semtacisi(nx2,ny2,nx1,ny1)
92
93      alter control 9 value str$(deg1)
94      alter control 10 value str$(deg2)
95      alter control 11 value str$(round(sqr((ny2-ny1)^2+(nx2-nx1)^2),0.001))
96  end sub
97
98  sub yaz
99      dim ad1,ad2 as string
100     ad1=readcontrolvalue(1)
101     ad2=readcontrolvalue(4)
102     alter control 12 title chr$(40)+ ad1 + chr$(45) + ad2+chr$(41)
103     alter control 13 title chr$(40)+ ad2 + chr$(45) + ad1+chr$(41)
104  end sub
105
106  function semtacisi(Byval x1,y1,x2,y2 as float) as float
107     dim pi as float
108     pi=3.14159265358979
109     if (y2-y1)>0 and (x2-x1)>0 then
110         semtacisi=atn((y2-y1)/(x2-x1))*(200/pi)
111     ElseIf (y2-y1)>0 and (x2-x1)<0 then
112         semtacisi=200 - (atn((y2-y1)/(x2-x1))*(-1))*(200/pi)
113     ElseIf (y2-y1)<0 and (x2-x1)<0 then
114         semtacisi=200 + (atn((y2-y1)/(x2-x1))*(200/pi))
115     Else
116         semtacisi=400 - (atn((y2-y1)/(x2-x1))*(-1))*(200/pi)
117     End if
118  end function

```


Şekil 71

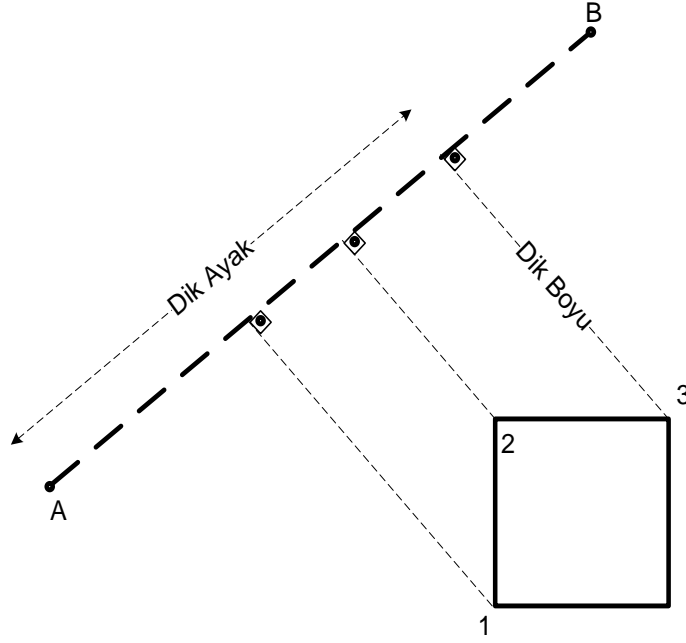
RadioGroup Kullanımı ve Yan Nokta Hesabı

Şekil 73, Şekil 74, Şekil 75, Şekil 76 ve Şekil 77 aynı uygulamanın kod örneğidir. Uygulamada yan nokta hesabı örneği bulunmaktadır. Şekil 72 yan nokta kavramının anlaşılması için yapılan tasvirdir. Şekil 72 üzerindeki 2 ve 3 numaralı noktalar yan nokta olarak ifade edilen noktalardır. P.7 ve P.9 güzergah üzerindeki başlangıç ve bitiş noktaları olarak düşünülmelidir. 2 ve 3 numaralı noktaların koordinatlarının bulunması için, başlangıç noktasından itibaren güzergah üzerindeki dik ayak uzunluğu ve dik ayak uzunluğunun bittiği noktadan itibaren yan noktalara olan dik boy uzunluğu gereklidir. Şekil 72 örneğine göre, P.7 başlangıç noktası ve P.9 bitiş noktası olmak üzere, 2 numaralı yan nokta için 12.456 m dik ayak; 4.992 m ise dik boy uzunluğudur. Yan nokta koordinatlarının hesaplanması için başlangıç ve bitiş noktaları arasındaki (P.7 ile P.9) arasındaki semt açısının hesaplanması da gereklidir.



Şekil 72

Uygulamada *kontrol*, *main*, *temizle* ve *yannokta* isimli yordamlar ile *semt_hesapla* isimli fonksiyon bulunmaktadır. Main yordamı (ilk çalışan yordam) içerisinde, daha önceki örneklerde olduğu gibi, form tasarımına ait kod bulunmaktadır. *kontrol* isimli yordamda kullanıcının veri girişi yapıp yapmadığının kontrolü ve kullanıcıdan alınan verilerin *yannokta* yordamına eklenmesi işlemleri yapılmaktadır. Şekil 75 114 numaralı satırda *yannokta* yordamına veri eklenmesi ve *call* komutu ile *yannokta* yordamının çağırılması kod satırı bulunmaktadır.



Kod içerisinde *semt_hesapla* fonksiyonu ile başlangıç ve bitiş noktaları arasındaki semt açısının hesaplanması için kullanılmıştır. Şekil 76 126 ile 138 numaralı satırlarda *semt_hesapla* fonksiyonu kod bloğu bulunmaktadır. Fonksiyon verilen koordinat değerlerine göre, semt açısının hangi bölge içinde kaldığı ve bölgeye göre semt açısının hesaplanmasını sağlamaktadır. Fonksiyon kod içerisinde Şekil 75 113 numaralı satırda çağırılmakta, fonksiyonun geri döndürdüğü değer *semtdeg* isimli değişkene aktarılmaktadır. Fonksiyon tanımında, geriye döndüreceği değer veritipi:

```
Function semt_hesapla(byval x1, x2, y1, y2 as float) as float
End function
```

Kırmızı ve kalın nitelenen *as float* ifadesi belirtilmiştir. *semt_hesapla* fonksiyonunun geriye döndürdüğü değer tutulduğu *semtdeg* değişkeni yannokta yordamı içine parametre olarak eklenmiştir (Şekil 75 114 numaralı satır).

Şekil 77 program çalıştırıldığında kullanıcının karşına çıkacak olan ara yüz örneğidir. Kullanıcı başlangıç ve bitiş noktalarının koordinatlarını, koordinatı hesaplanacak nokta için dik ayak ve dik boy değerlerini girdikten sonra noktanın güzergahın solunda veya sağında olduğunu belirtmesi gerekmektedir. Çünkü noktanın güzergahın sağında veya solunda olmasına göre hesaplama formülleri farklıdır. Bu işlemin yapılması için *RadioGroup* kontrolü kullanılmıştır. *RadioGroup* kontrolü tanımı Şekil 74 içinde 52 ile 56 numaralı satırlar arasında yapılmıştır. Tanımlanan *RadioGroup* kontrolünün Id değeri 7 olarak tanımlanmıştır. yannokta yordamı içinde kullanıcının, *RadioGroup* radyo düğmelerinden *Nokta Solda* veya *Nokta Sağda* (Şekil

77) seçimi sınanmıştır (Şekil 76 142 ile 148 numaralı satırlar arasındaki if – then bloğu içinde). Kullanıcının seçtiği radyo düğmesine göre, yordamdaki parametreler ilgili denklem içine aktarılmıştır.

Şekil 76 149 numaralı satırda temizle yordamı ile hesaplama yapıldıktan sonra kullanıcının girdiği koordinat değerleri, dik ayak ve dik boy değerleri temizlenmektedir.

Şekil 76 150 ve 151. satırlarda, yan noktanın hesaplanan koordinat değerleri 8 ve 9 id değerlerine sahip kontrollere *Alter* komutu ile aktarılmıştır.

Yan nokta Solda ise kullanılacak formül:

$$nokY = Y_{BAŞ} + dikayak * \sin(semtBAŞ - SON) - dikboy * \cos(semtBAŞ - SON)$$

$$nokX = X_{BAŞ} + dikayak * \cos(semtBAŞ - SON) + dikboy * \sin(semtBAŞ - SON)$$

Yan nokta Sağda ise kullanılacak formül:

$$nokY = Y_{BAŞ} + dikayak * \sin(semtBAŞ - SON) + dikboy * \cos(semtBAŞ - SON)$$

$$nokX = X_{BAŞ} + dikayak * \cos(semtBAŞ - SON) - dikboy * \sin(semtBAŞ - SON)$$

```
1 include "mapbasic.def"
2 declare sub kontrol
3 declare sub main
4 declare sub temizle
5 declare function semt_hesapla(byval x1,x2,y1,y2 as float) as float
6 declare sub yannokta(byval x1,x2,y1,y2,dik_ayak, dik_boy,semt_aci as float)
7 sub main
8 Dialog
9     Title "Yan Nokta Hesabı"
10    Width 305 Height 122
11
12 Control StaticText
13     Position 8, 27
14     Width 63 Height 8
15     Title "Başlangıç Noktası"
16
17 Control EditText
18     Id 1
19     Position 75, 25
20     Width 62 Height 12
21
22 Control EditText
23     Id 2
24     Position 141, 25
25     Width 62 Height 12
26
27 Control EditText
28     Id 3
29     Position 75, 48
30     Width 62 Height 12
31
32 Control StaticText
33     Position 9, 48
34     Width 44 Height 8
35     Title "Bitiş Noktası"
36
37 Control EditText
38     Id 4
39     Position 141, 48
40     Width 62 Height 12
```

Şekil 73

```
42 Control StaticText
43     Position 90, 16
44     Width 33 Height 8
45     Title "Y Değeri"
46
47 Control StaticText
48     Position 156, 16
49     Width 33 Height 8
50     Title "X Değeri"
51
52 Control RadioGroup
53     Id 7
54     Position 207, 25
55     Title "Nokta Solda; Nokta Sağda"
56     Value 1
57
58 Control Button
59     Title "Hesapla"
60     Position 126, 77
61     Width 51 Height 14
62     calling kontrol
63 Control StaticText
64     Position 185, 69
65     Width 46 Height 8
66     Title "Yan Nokta Y"
67
68 Control StaticText
69     Position 185, 87
70     Width 46 Height 8
71     Title "Yan Nokta X"
72
73 Control EditText
74     Id 8
75     Position 235, 69
76     Width 62 Height 12
77
78 Control EditText
79     Id 9
80     Position 235, 86
81     Width 62 Height 12
```

Şekil 74

```
83 Control StaticText
84     Position 9, 69
85     Width 34 Height 8
86     Title "Dik Ayak"
87
88 Control StaticText
89     Position 9, 87
90     Width 30 Height 8
91     Title "Dik Boy"
92
93 Control EditText
94     Id 5
95     Position 48, 69
96     Width 62 Height 12
97
98 Control EditText
99     Id 6
100    Position 48, 86
101    Width 62 Height 12end sub
102 sub kontrol
103    dim x1,y1,x2,y2,dik_ayak,dik_boy,semtdeg as float
104    if readcontrolvalue(1)<>" " and readcontrolvalue(2)<>" " then
105        if readcontrolvalue(3)<>" " and readcontrolvalue(4)<>" " then
106            if readcontrolvalue(5)<>" " and readcontrolvalue(6)<>" " then
107                y1=readcontrolvalue(1)
108                x1=readcontrolvalue(2)
109                y2=readcontrolvalue(3)
110                x2=readcontrolvalue(4)
111                dik_ayak=readcontrolvalue(5)
112                dik_boy=readcontrolvalue(6)
113                semtdeg=semt_hesapla(x1,x2,y1,y2)
114                call yannokta(x1,x2,y1,y2,dik_ayak, dik_boy,semtdeg)
115            else
116                note "Başlangıç Noktasının Koordinatlarını girmelisiniz!"
117            end if
118        else
119            note "Bitiş Noktasının Koordinatlarını girmelisiniz!"
120        end if
121    else
122        note "Başlangıç Noktasının Koordinatlarını girmelisiniz!"
123    end if
```

Şekil 75

```
125 end sub
126 function semt_hesapla(byval x1,x2,y1,y2 as float) as float
127     dim pi as float
128     pi=3.14159265358979
129     if (y2-y1)>0 and (x2-x1)>0 then
130         semt_hesapla=atn((y2-y1)/(x2-x1))*200/pi
131     elseif (y2-y1)>0 and (x2-x1)<0 then
132         semt_hesapla=200 - atn((y2-y1)/(x2-x1)*(-1))*200/pi
133     elseif (y2-y1)<0 and (x2-x1)<0 then
134         semt_hesapla=200 + atn((y2-y1)/(x2-x1))*200/pi
135     else
136         semt_hesapla=400 - atn((y2-y1)/(x2-x1)*(-1))*200/pi
137     end if
138 end function
139 sub yannokta(byval x1,x2,y1,y2,dik_ayak, dik_boy,semt_aci as float)
140     dim nok_y, nok_x,pi as float
141     pi=3.14159265358979
142     if readcontrolvalue(7)=1 then
143         nok_y=y1+dik_ayak*sin(semt_aci*pi/200)-dik_boy*cos(semt_aci*pi/200)
144         nok_x=x1+dik_ayak*cos(semt_aci*pi/200)+dik_boy*sin(semt_aci*pi/200)
145     else
146         nok_y=y1+dik_ayak*sin(semt_aci*pi/200)+dik_boy*cos(semt_aci*pi/200)
147         nok_x=x1+dik_ayak*cos(semt_aci*pi/200)-dik_boy*sin(semt_aci*pi/200)
148     end if
149     call temizle
150     alter control 8 value str$(nok_y)
151     alter control 9 value str$(nok_x)
152 end sub
153 sub temizle
154     alter control 1 value ""
155     alter control 2 value ""
156     alter control 3 value ""
157     alter control 4 value ""
158     alter control 5 value ""
159     alter control 6 value ""
160 end sub
```

Şekil 76

Yan Nokta Hesabı

Y Değeri X Değeri

Başlangıç Noktası Nokta Solda
 Nokta Sağda

Bitiş Noktası

Dik Ayak Yan Nokta Y

Dik Boy Hesapla Yan Nokta X

Şekil 77

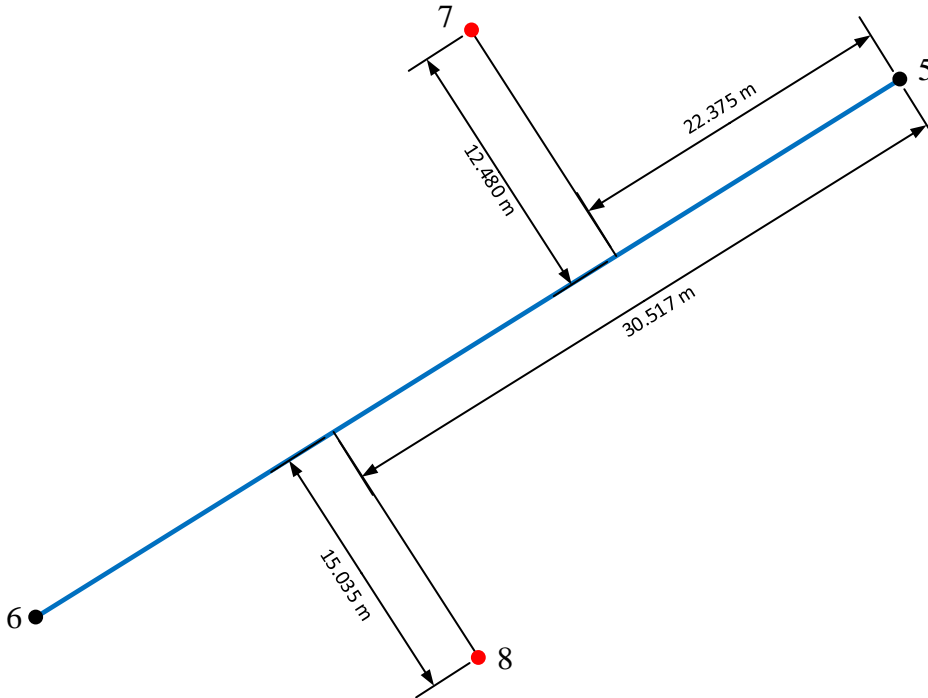
Tablo 7

NNo	Y	X
5	560001.854	4358276.791
6	559957.872	4358242.470

Tablo 8

NNo	Dik Ayak	Dik Boy
7	22.375 m	12.480 m
8	30.517	-15.035

5 numaralı nokta yol güzergahının başlangıç noktası olmak üzere, yolun sağında ve solunda belirlenen noktalarda elektrik direkleri dikilmek isteniyor. Elektrik direklerinin dikeleceği noktaların mesafeleri Tablo 8’de verilmiştir. Tablo 7’de güzergâhın başlangıç ve bitiş noktalarının koordinatları verilmiştir. Elektrik direklerinin koordinatlarını hesaplayınız.



Döngü

Yapılacak uygulamada aynı kod satırlarının benzer parametreler kullanılarak birden fazla kullanılması gerektiği takdirde döngü yapıları kullanılır. Döngü yapılarında, istenen kod bloğunun tekrar sayısı, ya program kodunu yazan kullanıcının belirleyeceği sayıda sabittir ya da program içinde verinin okunduğu (çekildiği) dosya veya veri tabanı kayıt sayısına göre değişkendir. Döngü yapıları için farklı komut yapıları vardır.

For – Next Döngü Yapısı

For – Next döngüsünde döngünün tekrar sayısını belirlemek için başlangıç değeri değişken parametresiyle kullanılır (ifade 8). Değişkene ilk aldığı değer atanır (ifade 8 başlangıç değeri), to ifadesinden sonra da bitiş değeri belirlenir. Başlangıç değerinden itibaren artış miktarı (aksi belirtilmedikçe) +1 değeri kadar olacaktır. Döngü içindeki program kod bloğunun bitişi Next ifadesi eğer değişken bitiş değerine ulaşmadıysa next ifadesinden başa dönlür.

ifade 8

For *değişken* = başlangıç **to** bitiş
Döngüde kullanılacak komutlar
next

Örnek:

dim i as integer

for i = 1 **to** 5 → i tanımlı değişken ve değişkene 1 atanmış, döngü bitişi 5 verilmiş

print “merhaba” → döngü içinde print ekranına “merhaba” yazılacak

Next → döngü kod bloğunun bitişini ve döngü tekrar sayısının bitişini belirleyen ifade

Örnek içindeki çalışma prensibi:

i değeri	Kod bloğu
i 1 değerini alır	Print ekranına “merhaba” yazılır, next ifadesinde i değerinin 5 olup olmadığı kontrol edilir. 5 değilse başa döner, i değeri 1 artar
i 2 değerindedir	Print ekranına “merhaba” yazılır, next ifadesinde i değerinin 5 olup olmadığı kontrol edilir. 5 değilse başa döner, i değeri 1 artar
i 3 değerindedir	Print ekranına “merhaba” yazılır, next ifadesinde i değerinin 5 olup olmadığı kontrol edilir. 5 değilse başa döner, i değeri 1 artar
i 4 değerindedir	Print ekranına “merhaba” yazılır, next ifadesinde i değerinin 5 olup olmadığı kontrol edilir. 5 değilse başa döner, i değeri 1 artar

i	5	Print ekranına “merhaba” yazılır, next ifadesinde i değerinin 5 olup olmadığı kontrol edilir. Eğer i 5 ise döngüden <u>çıkılır</u>
---	---	---

Do – Loop Döngü Yapısı

Do loop döngüsünde, döngünün sabit artış değerinden ziyade, belirtilen sınama (koşul) sağlandığı sürece döngü tekrarlanır. Sınama işlemi ya döngünün başlangıcında yapılır ya da döngünün sonunda yapılır. Sınama işlemi cümle bazlı düşünülürse, “sınama sağlandığı sürece” (while operatörü kullanılarak) döngü sağlanır veya “sınama sağlanana kadar” (until operatörü kullanılarak) döngü sağlanır. Until veya while operatörleri döngünün başında (do satırında) ya da döngünün sonunda (Loop satırında kullanılır). Do – Loop döngüsü metin dosyalarından veri okumak veya veri tabanındaki tablolardan veri okumak için elverişlidir.

ifade 9

Do [until yada while] (sınama)

(Döngü içindeki kod bloğu)

Loop [until yada while]

Örnek: Do – Loop döngüsü kullanılarak ekrana 3 defa “Kaman MYO” yazınız

ifade 10 ve ifade 11 *Do – Loop* döngüsünde while ve until yapılarının kullanımına örnektir.

ifade 10

```
Dim i as integer
i = 1
Do while (i < 3)
    print "Kaman Myo"
    i = i + 1
Loop
```

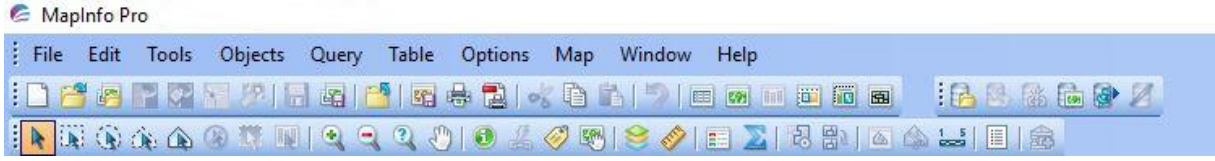
ifade 11

```
Dim i as integer
i = 0
Do
    print "Kaman Myo"
    i = i + 1
Loop until (i < 3)
```

Buton (Düğme) Araçları

Butonlar (düğmeler) yazılımlarda kullanıcıların yapacağı işlemlerde kolaylık sağlayan, işleme yön verilmesini sağlayan, düğmenin kendisine yüklü bir olayın veya bir fonksiyonun olduğu araçlardır. Şekil 78 MapInfo yazılımında var olan butonların resmi bulunmaktadır. Her buton tanımlanmış bir fonksiyonu, bir işlemi veya bir analizi yerine getirmektedir. Butonlar menülerden farklı olarak, tasarlanan olaya dayalı bir kullanım amaçlıdır. Menüye tıkladığımızda

bir pencere açılabilir. Ama pencerenin açılması sürekli çizimi veya istenilen fonksiyonu sağlamaz veya menü bir noktayı seçtikten sonra noktanın koordinatlarının gösterimi için uygun olabilir ama ardı ardına seçilen noktaların koordinatlarını göstermek için çok uygun olmazlar.



Şekil 78

Butonları oluşturmak için *Dialog* formu gibi bir form oluşturmamız gerekli. Butonlar için oluşturulacak yüzeye *ButtonPad* denir.

Create ButtonPad “Başlık” As	ButtonPad tanımı yapıldıktan sonra butonlar eklenebilir.
{PushButton ToggleButton ToolButton }	üç çeşit buton tipi vardır. Bunlardan biri yazılır
Calling yordam_adı	butona tıklandığında ilgili yordam veya fonksiyonu çağılır
ID buton_id_deger	Butona verilen tekil numara
Icon ilgili ikon	butonun üzerinde gözükecek işlemi niteleyen simge / ikon
Cursor n	Tıklandığında fare imlecinin nasıl olacağı belirtilir. Sadece Toolbutton tipinde çalışır.
DrawMode	kullanıcının çizim ya da seçim modlarından hangisini yapacağını belirtir.
HelpMsg “Yardım mesajı”	kullanıcının buton üzerine geldiğinde gözükecek yardım mesajı
Enable	butonun görünür ve aktif kullanılabilir durumda olması
Disable	butonun görünür fakat pasif kullanılamaz durumda olması
Check	butonun devamlı seçili vaziyette olması
Uncheck	butonun devamlı seçili olmaması vaziyetinde olması

Buton Tipleri Örneği

Şekil 79 Buton tipleri PushButton, ToolButton, ToggleButton üç ayrı buton tipi için oluşturulmuş örnek program kodunu içerir. Form üzerindeki buton (düğme) kontrolünden ayrı olarak bir taşıyıcı pad üzerinde tanımlanmış düğmelerden bahsedilmektedir. Bu araçlar harita penceresinde kullanılan düğmelerdir. Harita penceresinde düğmeye tıklandığında bir olay oluşmuş olacak. Olay ile bir yordam veya fonksiyon çağırılabilir. Bu sayede düğmeye uygulama amaçlı bir görev verilmiş olacaktır.

```

1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub pussbutton_sub
5 declare sub togglebutton_sub
6 declare sub toolbutton_sub
7 Sub Main()
8   I Create ButtonPad "Yardımcı" As
9     PushButton
10      Calling pussbutton_sub
11      Icon MI_ICON_ZOOM_QUESTION
12     ToolButton
13      Calling toolbutton_sub
14      Icon MI_ICON_CROSSHAIR
15      DrawMode DM_CUSTOM_LINE
16     ToggleButton
17      Calling togglebutton_sub
18      Icon MI_ICON_RULER
19      Check
20     Title "Yardım"
21     width 50
22     show
23 End Sub
24 sub pussbutton_sub
25   note "deneme pussbutton"
26 End Sub
27 sub togglebutton_sub
28   note "deneme togglebutton"
29 End Sub
30 sub toolbutton_sub
31   note "deneme toolbutton"
32 End Sub

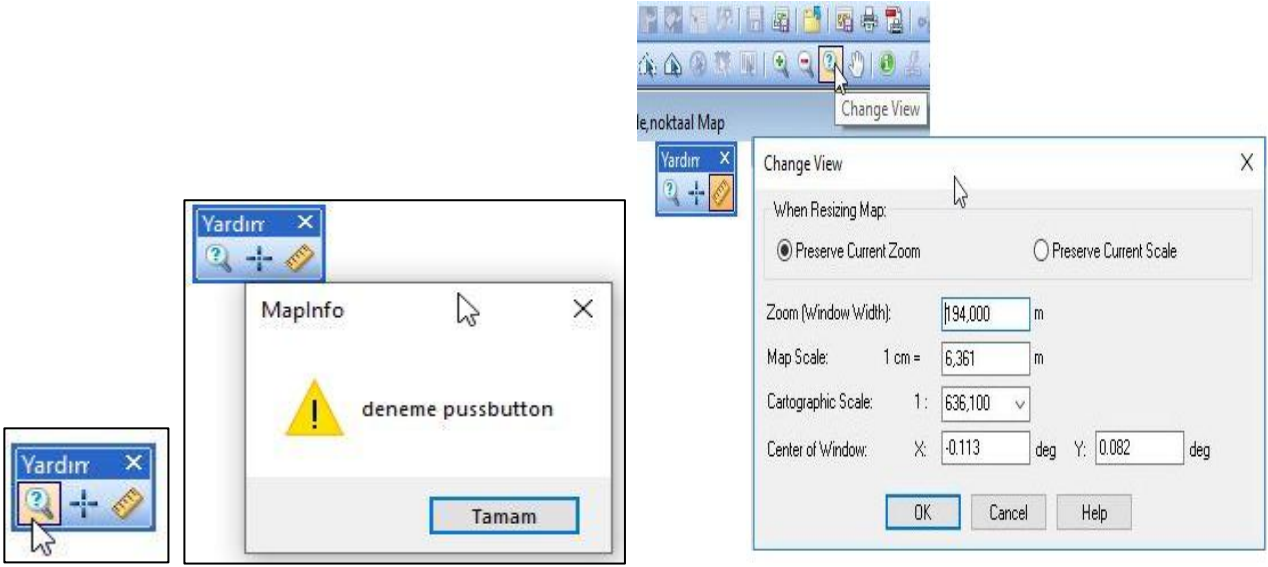
```

Şekil 79



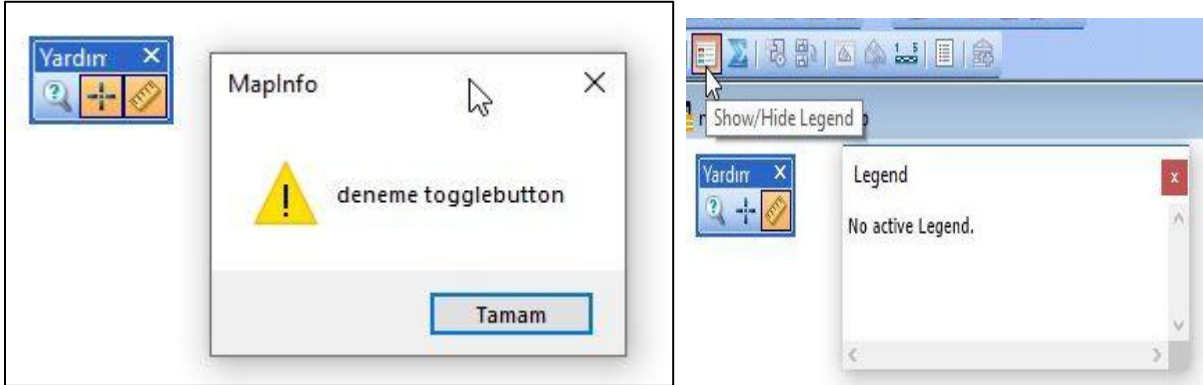
- 1 Numara: PushButton
- 2 Numara: ToolButton
- 3 Numara: ToggleButton

1 Numaralı PushButton kullanıcının genelde bir menü çalıştırmak, bir yordam çalıştırmak ya da bir pencere açmak gibi özelliği vardır. MapInfo penceresinde *Change View* (Görünümü değiştir) butonu buna örnektir (Şekil 80 En sağ resim).



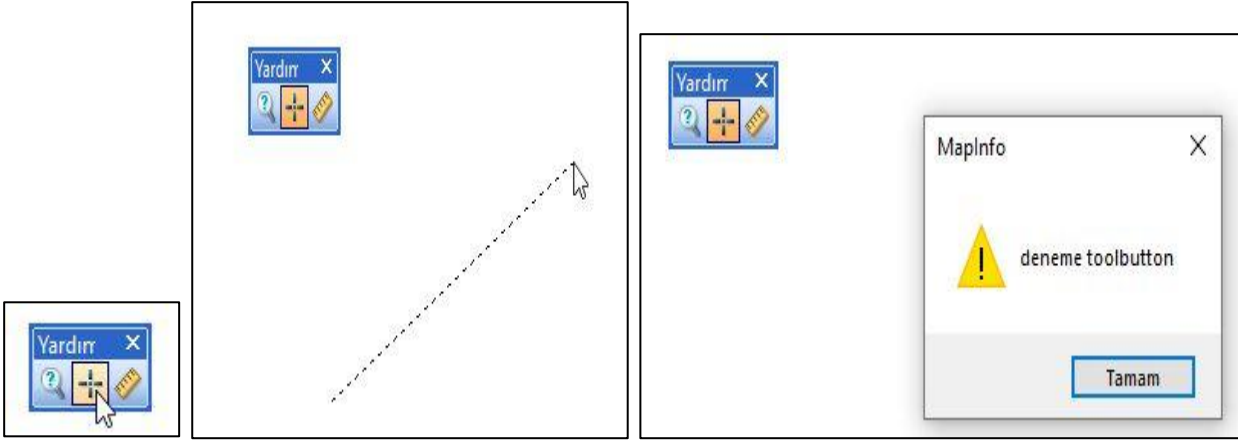
Şekil 80

3 Numaralı *Togglebutton* MapInfo penceresinde *Legend* (lejant) ve yanındaki istatistik butonu gibi ikinci kere tıklanıncaya kadar aynı işlemi yaparlar (Şekil 81 sağ resim). Örneğin Legend butonu tekrar tıklanıncaya kadar Legend penceresi ekranda gözüktür. Şekil 81 sol resim de cetvel sembolü olan düğme togglebutton örneğidir.



Şekil 81

2 Numaralı *ToolButton*, tıkladığında aktif kalır ve başka bir *toolbutton* seçilene kadar aktifliği bozulmaz. Genellikle seçim ve çizim araçları için harita ve çıktı penceresinde kullanılır. Şekil 82 çizgi çizimi örneği için kullanılan *toolbutton* örneğidir.



Şekil 82

Haritada Tıklanan Yerin Koordinatlarını Ekrana Yazan Buton Programı

```

koordinatyaz.mb
1  include "mapbasic.def"
2  include "icons.def"
3  declare sub main
4  declare sub koord_yaz
5
6  sub main
7
8      create buttonpad "SOR" as
9          toolbar calling koord_yaz
10         ICON MI_ICON_LETTERS_K
11
12  end sub
13
14  sub koord_yaz
15      dim x,y as float
16      dim i,k as integer
17      dim tabname as string
18
19      set coordsys window frontwindow()
20      x=commandinfo(cmd_info_x)
21      y=commandinfo(cmd_info_y)
22      print chr$(12)
23      print "X :"+y+"      Y :"+x
24
25
26
27  end sub

```

Şekil 83

Şekil 83 Mouse

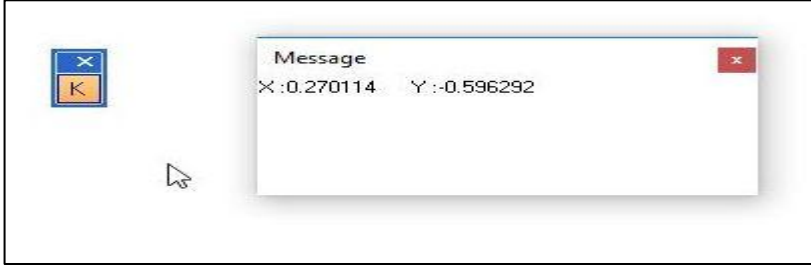
ile harita ekranında tıklanılan yerin (noktanın) koordinatlarını veren bir buton uygulaması örneğidir. Örnekte *ToolButton* tipinde buton kullanılmıştır. Yani bir kere butona tıkladığında farklı bir Toolbuttona tıklanana kadar işlem sürecektir. Oluşturulan butonun üzerinde K harfi, sembol olarak gözükecektir (Şekil 44 10. satır). Butona tıkladığında *koord_yaz* isimli yordam çalışıyor.

Satır 19 *set*

coordsys komutları yapılacak işlemin koordinat sistemini belirtmek için kullanılıyor. *Window frontwindow()* komutlar en öndeki pencere bilgisi için kullanılıyor.

set coordsys Window frontwindow() komut satırı en öndeki pencerenin koordinat sistemine göre Mapbasic programında yapılan işlemlerin koordinatlarının ayarlanmasını ifade ediyor. Set coordsys komutu ile haritanın koordinat sistemini ayarlayamazsınız. Ayarlamak için set map coordsys kullanmanız gereklidir.

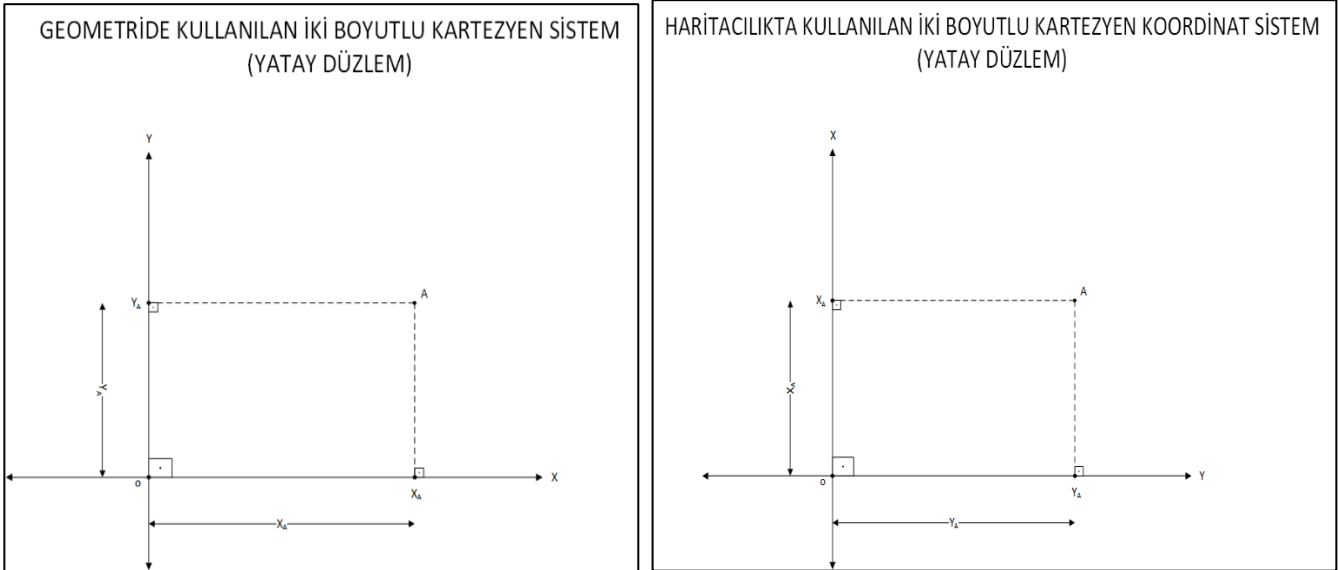
Commandinfo(cmd_info_x) yapılan seçime ait X koordinatını veriyor. *Print* komutu ekranda bir bilgi penceresi açar. *Note* komutuyla açılan bilgi penceresinden farklı olarak ekranda devalı bilgi değişiminin yapılabildiği bir bilgi penceresidir. *Chr\$(12) print* komutu ile açılan pencerenin ekranının temizlenmesi için kullanılır.



Şekil 84



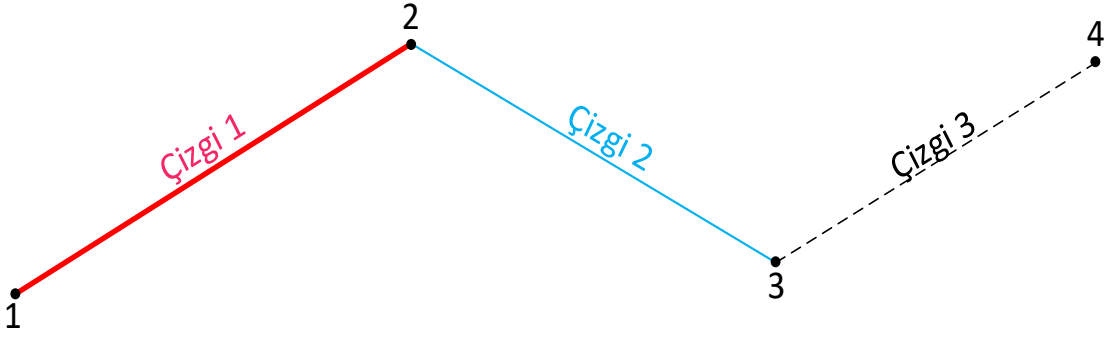
Ülkemizde harita yapımı amacıyla kullanılan X-Y yatay düzleminde, X yukarı doğru Y eksenini ise sağa doğru artırır. Mapinfo gibi yabancı menşeli yazılımlarda X-Y eksenlerinin artış yönleri aynı geometride olduğu gibidir.



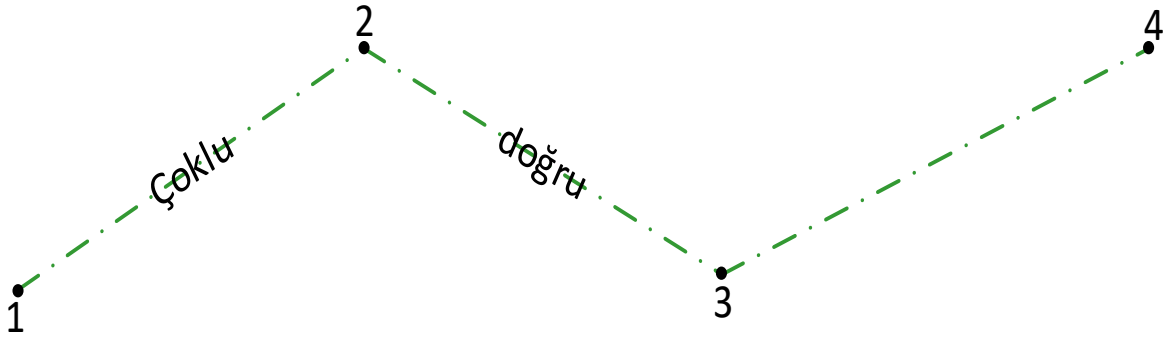
Şekil 85

Çizgi veya Çoklu Doğru Grafik Objelinin Uzunluk Bilgisi

Çizgi ve çoklu doğru objeleri birbirinden farklı objelerdir. Çizgi grafik objesi tek bir parçadan oluşur ve peşi sıra durmaksızın çizilmiş çizgiler bütünüünün her parçası kendi başına ayrı objedir (Şekil 86). Çoklu doğru grafik objesi birden fazla parçadan oluşur ve tüm parçaların birleşimi tek bir objedir (Şekil 87).



Şekil 86



Şekil 87

Her iki obje de kırıldıkları kısımlardaki (Vertex) noktalardan oluşur (Şekil 86 ve Şekil 87’de noktalar). Çizgi veya çoklu doğru objelerinin ad bilgileri dışında uzunlukları kırılma noktaları kullanılarak veya oluşturuldukları yazılım içinde obje özelliklerinden yararlanılarak hesaplanır.

Mapbasic programlama dilinde birden fazla fonksiyon çizgi veya çoklu doğru objelerinin uzunluklarını hesaplamak için kullanılır. Bu fonksiyonlar objectlen(), distance(), cartesiandistance(), sphericalobjectlen() ve sphericalobjectdistance().

Cartesiandistance() Fonksiyonunun Kullanımı

Cartesiandistance() fonksiyonu içerisine 5 parametre alır. İlk iki koordinat birinci noktanın X ve Y koordinatları, 3. ve 4. parametreler ikinci X ve Y koordinatları, sonuncu parametre ise uzunluk değerinin birimidir (ifade 12).

ifade 12

$$\text{Cartesiandistance}(x1, y1, x2, y2, \text{"birim"})$$

Cartesiandistance() fonksiyonu sadece kartezyen koordinat değerleri (X ve Y yatay düzlem koordinatları) kullanarak hesaplama yapar. Eğer X ve Y değerlerine coğrafi (veya jeodezik) enlem ve boylam değerleri girilirse geriye -1 değeri döndürür.

Şekil 88 Cartesiandistance() fonksiyonu kullanımına bir örnektir. Kullanıcının Mouse ile ekrana ilk tıklamasında birinci noktanın koordinatları elde edilir, ikinci tıklamasıyla da ikinci noktanın koordinatları ekrandan çekilir. İki noktanın koordinatları Cartesiandistance() fonksiyonuna aktarılır ve sonuç print ekranına yazdırılır.

Tablo 9 Cartesiandistance() fonksiyonu 5. Parametresinde yazılan uzunluk birim değerleri ve tanımlarıdır.

Tablo 9 (Pitney Bowes, 2014)

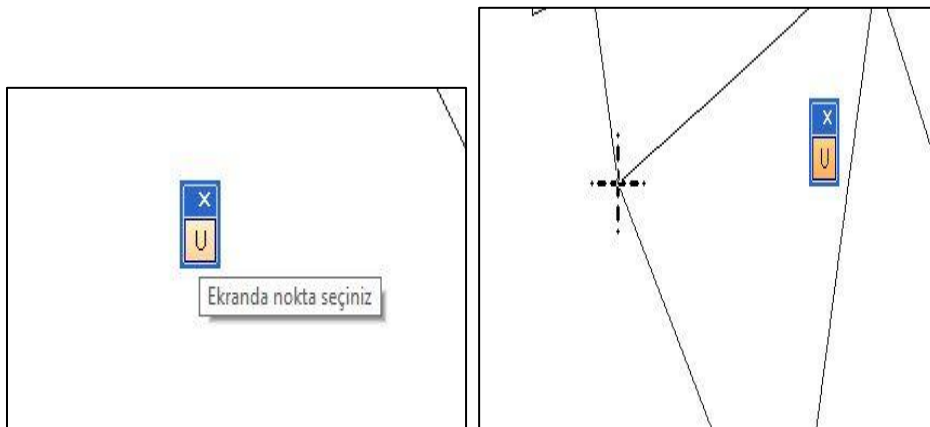
Birim Adı	Birim Tanımı
"ch"	Chain (1 ch = 20.1168 m)
"cm"	santimetre
"ft"	Feet veya Türkçe'de fit olarak ifade edilir 1 fit 30.48 cm'e eşittir
"in"	İnç. 1 in = 0.0254 m
"km"	kilometre
"li"	Link, Türkçe'de bağ olarak da ifade edilir. 1ln = 0.2011684023368 m
"m"	metre
"mi"	Mil, 1 mi = 1609.344 m
"mm"	millimetre
"nmi"	Deniz mil değeri olarakda bilinir. (1 deniz mili yaklaşık 1852 metre) 1nmi= 1852 m
"rd"	Rod, 1 rod= 5.0292 m
"survey ft"	Amerikda'da harita yapımı için gerekli ölçümlerde kullanılan bir birim. 1 suvey ft = 30.48006 cm
"yd"	Yard, 1 yd= 0.9144 m

```

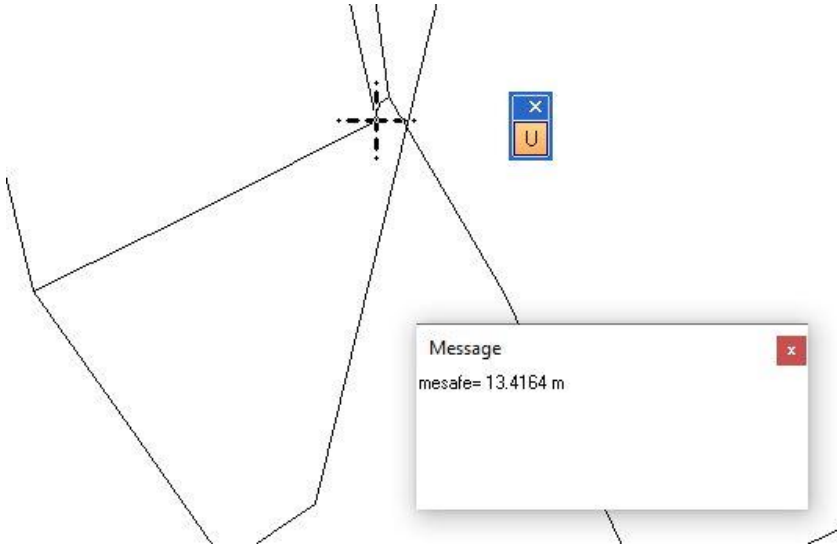
1 include "mapbasic.def"
2 include "icons.def"
3 declare sub main
4 declare sub uzunluk_sor
5 dim i,say,x1,y1,x2,y2 as integer
6 sub main
7     i=0
8     create buttonpad "Sor" as
9         |         |         |
10        |         |         |         |
11        |         |         |         |         |
12        |         |         |         |         |         |
13        |         |         |         |         |         |
14        |         |         |         |         |         |
15        |         |         |         |         |         |
16        |         |         |         |         |         |
17        |         |         |         |         |         |
18        |         |         |         |         |         |
19        |         |         |         |         |         |
20        |         |         |         |         |         |
21        |         |         |         |         |         |
22        |         |         |         |         |         |
23        |         |         |         |         |         |
24        |         |         |         |         |         |
25        |         |         |         |         |         |
26        |         |         |         |         |         |
27        |         |         |         |         |         |
28        |         |         |         |         |         |
end sub
sub uzunluk_sor
dim mesafe as float
set coordsys window frontwindow()
i=i+1
if i=1 then
    x1=commandinfo(cmd_info_x)
    y1=commandinfo(cmd_info_y)
elseif i=2 then
    i=0
    x2=commandinfo(cmd_info_x)
    y2=commandinfo(cmd_info_y)
mesafe=cartesiandistance(x1,y1,x2,y2,"m")
print chr$(12)
print "mesafe= " + mesafe + " m  i=" + i
end if
end sub

```

Şekil 88



Şekil 89



Şekil 90

Distance(x1,y1,x2,y2, birim) Fonksiyonu:

Kullanımı Cartesiandistance() fonksiyonuna benzer, Cartesiandistance() fonksiyonundan farklı olarak, koordinat olarak enlem – boylam değerleri kullanılabilir. Enlem boylam değerlerinin sıralamasında X yerine boylam, Y koordinatı yerine enlem değerleri girilir. Eğer koordinatlar enlem boylam olarak girildiyse, en kısa küresel eğri mesafe değeri verilir (loksodrom eğrisi).

CartesianObjectLen() fonksiyonu:

CartesianObjectLen() fonksiyonu iki birimi vardır. İlk parametre seçilen obje, 2. Paramtere ise uzunluk birimidir. Çizgi ve çoklu doğru objelerinin uzunluk değerlerini belirlemede kullanılır. İlk parametre olarak belirtilen parametre seçilen parametre veya tabloda kayıtlı olan parametre olabilir. Coğrafik veya jeodezik enlem – boylam koordinat değerleri için bu işlemi yapmaz. Eğer haritanın koordinat sistemi coğrafik veya jeodezik enlem – boylam koordinat değerlerinden oluşuyorsa fonksiyon geriye -1 değeri döndürür. Şekil 91 Cartesianobjectlen() fonksiyonu kullanım örneği vardır (Şekil 91 39. satır). Örnekte, çizgi veya çoklu doğru objesi dışında bir obje seçilmiş olma olasılığı, bir obje seçilip seçilmediği gibi kontroller de yapılmaktadır. Seçilen objenin bulunduğu tablo (tabaka) üzerinden okunması sağlanıyor. İşlemlerin yapılabilmesi için bazı MapBasic.def tanım dosyasında tanımlı fonksiyonlar da kullanılmıştır. Aşağıda bu fonksiyonların özellikleri açıklanmıştır.

Windowinfo()→ Belirtilen pencerenin harita penceresi, grafik, tablo, düğme gibi bir obje olup olmadığı hakkında sayısal değer döndürüyor. MapBasic.def dosyası içinde geriye döndürdüğü değerlere dair liste bulunmakta.

Commandinfo() → Mouse ile harita ekranında tıklanılan noktadaki X-Y koordinatlarını almak için kullanılıyor.

Searchpoint() → Belirtilen ekranda (frontwindow() ile ekran numarası alınabilir) ve belirtilen koordinatlarda harita objesi olup olmadığını kontrol ediyor, geriye obje sayısı döndürüyor. Obje yoksa 0 (sıfır) döndürüyor.

Searchinfo() → Searchpoint() fonksiyonu ile beraber kullanılır. Searchpoint() tıklanılan noktada kaç obje olduğunun bulunması için kullanılır. Searchinfo(), searchpoint() fonksiyonun geriye döndürdüğü obje sayısını dikkate alarak, objenin kayıtlı olduğu tablonun (tabakanın) adını, objenin tablo içindeki kayıtlı olduğu satır sayısını döndürmek için kullanılır.

Fetch rec from → belirtilen tabloda (tabakadaki) belirtilen satırdaki tablo verisine (veya grafik obje verisine) erişmek için kullanılır.

Objectinfo() → Fonksiyon içine aldığı obje hakkında bilgi almak için kullanılır. Örnekte grafik obje türünü öğrenmek için kullanılmıştır. Program kodu incelendiğinde objectinfo() fonksiyonu hem çizgi hem de çoklu doğru grafik objeleri için işlemin yapılması amaçlı program kodu tasarlanmıştır.

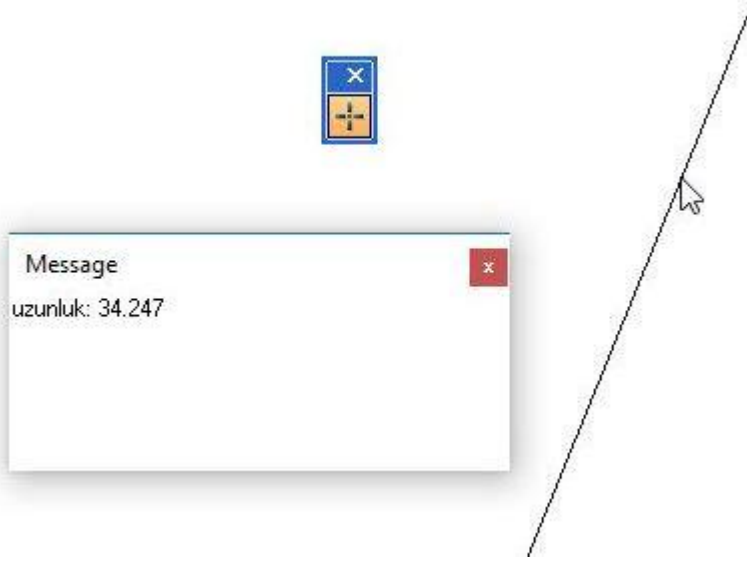
```

1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub uzunluk_bul
5 Sub Main()
6     create buttonpad "Cizgi" as
7         toolbutton
8             calling uzunluk_bul id 1
9             helpmsg "Çizgi Seç\nÇizgi Seç"
10 End Sub
11
12 sub uzunluk_bul
13     dim i, kayit_deger, obje_tip, pencere_id as integer
14     dim tablo_ad as alias
15     dim x_tık, y_tık as float
16     dim uzunluk as float
17     dim obje as object
18     set coordsys window frontwindow()
19     pencere_id=frontwindow()
20     if windowinfo(pencere_id,WIN_INFO_TYPE)<>WIN_MAPPER then
21         print chr$(12)
22         print "Bu araç harita penceresinde olmayan bir pencerede kullanılamaz"
23     else
24         x_tık=commandinfo(CMD_INFO_X)
25         y_tık=commandinfo(CMD_INFO_Y)
26         i=searchpoint(frontwindow(),x_tık,y_tık)
27         if i=0 then
28             print chr$(12)
29             Print "Seçtiğiniz Alanda grafik obje yok"
30         elseif i=1 then
31             tablo_ad=searchinfo(i,SEARCH_INFO_TABLE)
32             kayit_deger=searchinfo(i,SEARCH_INFO_ROW)
33             fetch rec kayit_deger from tablo_ad
34             tablo_ad=tablo_ad+".obj"
35             obje=tablo_ad
36             obje_tip=objectinfo(obje,OBJ_INFO_TYPE)
37             if obje_tip=OBJ_TYPE_LINE or obje_tip=OBJ_TYPE_PLINE then
38                 print chr$(12)
39                 print "uzunluk: " +Round(CartesianObjectLen( obje, "m" ),0.001)
40             else
41                 print chr$(12)
42                 print "Seçtiğiniz Objeye Doğru veya Çoklu Doğru objesi değil!"
43             end if
44         end if
45     end if
46 end sub

```

Şekil 91

Şekil 91 fare ile üzerine tıklanan çizgi objesinin Kartezyen uzunluğu olarak ifade edilen yatay mesafe değerini vermektedir. Daha önce üzerine fare ile tıklanan nokta objesinin koordinatlarının bulunmasında kullanılan program kodundan farklı olarak Şekil 91 37. Satırda objenin doğru (çizgi) veya çoklu doğru olup olmadığı sorgulanıyor. Ayrıca çizgi objesinin kartezyen uzunluğunu veren fonksiyon olan *CartesianObjectLen()* fonksiyonu kullanılıyor.



Çokgen grafik objesinin Alan Değerinin Hesaplanması

alanhesabi.mb

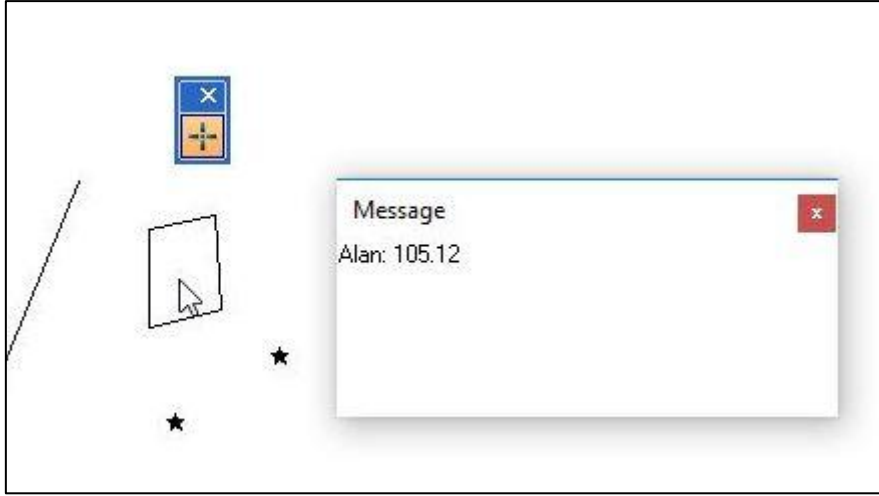
```

1  Include "MapBasic.def"
2  include "icons.def"
3  Declare Sub Main
4  declare sub alan_bul
5  Sub Main()
6      create buttonpad "Alan Bul" as
7          toolbar
8          calling alan_bul id 1
9          Helpmsg "Alan Seç\n Alan Seç"
10
11 End Sub
12
13 sub alan_bul
14     dim x_tik , y_tik as float
15     dim tablo_ad as alias
16     dim obje_tip , pencere_id, kayit_deger , i as integer
17     dim obje as object
18     set coordsys window frontwindow()
19     pencere_id=frontwindow()
20     if windowinfo(pencere_id,WIN_INFO_TYPE)<>WIN_MAPPER then
21         print chr$(12)
22         print "Bu araç harita penceresinde olmayan bir pencerede kullanılamaz"
23     else
24         x_tik=commandinfo(CMD_INFO_X)
25         y_tik=commandinfo(CMD_INFO_Y)
26         i=searchpoint(frontwindow(),x_tik,y_tik)
27         if i=0 then
28             print chr$(12)
29             Print "Bir obje seçmelisiniz"
30         elseif i=1 then
31             tablo_ad=searchinfo(i,SEARCH_INFO_TABLE)
32             kayit_deger=searchinfo(i,SEARCH_INFO_ROW)
33             fetch rec kayit_deger from tablo_ad
34             tablo_ad=tablo_ad+".obj"
35             obje=tablo_ad
36             obje_tip=objectinfo(obje,OBJ_INFO_TYPE)
37             if obje_tip=OBJ_TYPE_REGION then
38                 print chr$(12)
39                 print "Alan: "+ round( CartesianArea(obje, "sq m"),0.01)
40
41             else
42                 print chr$(12)
43                 print "Seçtiğiniz Objeye Alan objesi değil"
44             end if
45         end if
46     end if
47 end sub

```

Şekil 92

Şekil 92 üzerinde tıklanan alan değerini ekrana yazan program kodunu göstermektedir. Daha önceki nokta koordinatını gösterme ve çizgi uzunluğunu gösteren programlardan farklı olarak Şekil 92 satır 37 üzerinde tıklanan objenin alan objesi olup olmadığını sorgulanmaktadır. Bu işlem için üzerinde tıklanan objenin obje tipine dair sayısal değerin tutulduğu *obje_tip* değişkeninin *OBJ_TYPE_REGION* (Region= bölge, alan anlamındadır) değeriyle aynı olup olmadığı sorgulanıyor. İkinci farklı işlem ise alan değerini bulmak için *CartesianArea()* fonksiyonu kullanılmıştır. Fonksiyonun ilk parametresi üzerinde tıkladığımız obje, ikinci parametresi ise sonucun hangi birimde olduğu belirtir. Örnekte ikinci parametre “sq m” metre kare birimini ifade eder. Şekil 93 alan bulunmasına dair sonucun ekrana çıktısıdır.



Şekil 93

Haritada Seçilen Nokta Grafik Objesinin Koordinatlarını Yazdıran Program

```

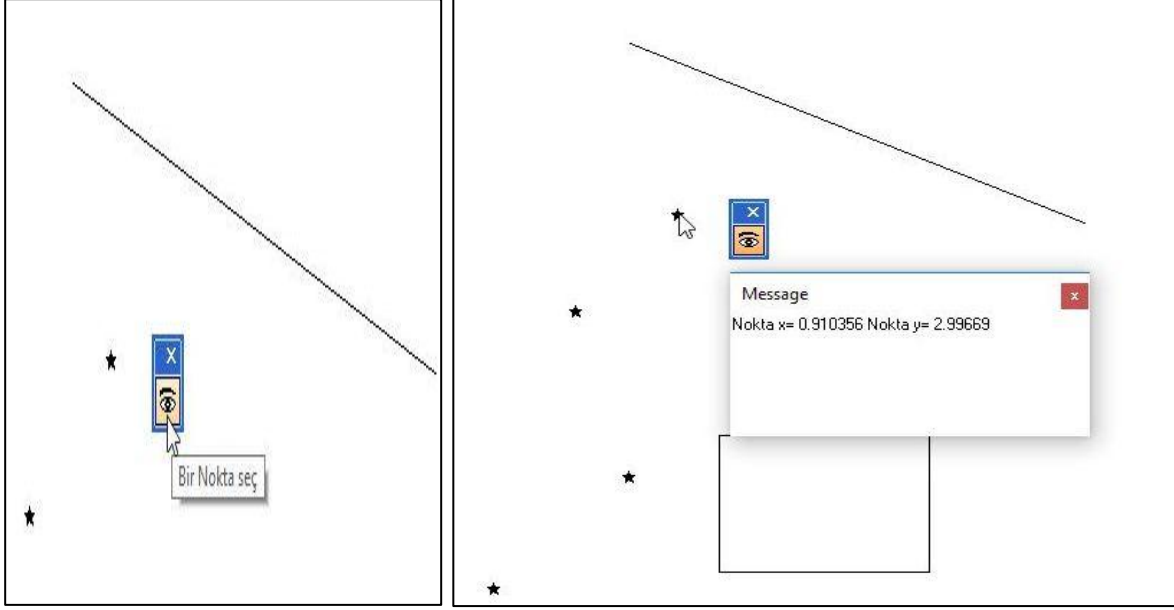
noktasecimvekoordinatgoster....
1  Include "MapBasic.def"
2  include "icons.def"
3  Declare Sub Main
4  declare sub nokta_koord_yaz
5
6  Sub Main()
7      create buttonpad "Nokta" as
8          toolbutton
9          calling nokta_koord_yaz id 1
10         icon MI_ICON_MISC_16
11         HelpMsg "Haritada Nokta Seç\nBir Nokta seç"
12
13 -End Sub
14
15 sub nokta_koord_yaz
16     dim x_tık, y_tık, x_obje, y_obje as float
17     dim kayıt_deger, i, obje_tip as integer
18     dim tablo_ad as alias
19     dim obje as object
20     Set Coordsys Window FrontWindow()
21     print chr$(12)
22     x_tık=commandinfo(CMD_INFO_X) 'Mouseun tıkladığı yerin X koordinatını verir
23     y_tık=commandinfo(CMD_INFO_Y) 'Mouseun tıkladığı yerin Y koordinatını verir
24     ' searchpoint haritadaki x,y koordinatindanki var olan tum tablolardaki tum nesnelere arıyor ve sayisini donduruyor
25     i = searchpoint( frontwindow(), x_tık, y_tık)
26     'eğer seçilen obje sayısı 0 ise seçim yapılmamıştır. Eğer 1 ise o tek objenin nokta olmasına göre koordinat yazılacak
27     if i=0 then
28         print "Bir Nokta Objesi Seçmelisiniz"
29     elseif i=1 then
30         'seçili objenin kayıtlı olduğu harita penceresi ve kayıt satır değerinin alınması
31         tablo_ad = SearchInfo(i, SEARCH_INFO_TABLE)
32         kayıt_deger = SearchInfo(i, SEARCH_INFO_ROW)
33         'harita penceresinin ilgili kaydına ulaşılır
34         fetch rec kayıt_deger from tablo_ad
35         'ilgili kayda ulaşıldıktan sonra grafik obje bilgisine ulaşabilmek için tablo_ad.obj
36         'ifadesini oluşturabilmek için alt satır kullanılır ve ilgili kayıttaki obje bilgileri bir objeye aktarılır
37         tablo_ad=tablo_ad+".obj"
38         obje=tablo_ad
39         'objenin obje tipini öğrenmek ve obje tipi nokta ise objenin koordinatlarının yazdırılması
40         obje_tip=objectinfo(obje,OBJ_INFO_TYPE)
41         if obje_tip=OBJ_TYPE_POINT then
42             x_obje = ObjectGeography(obje, OBJ_GEO_POINTX)
43             y_obje = ObjectGeography(obje, OBJ_GEO_POINTY)
44             print chr$(12)
45             print "Nokta x= " + x_obje + " Nokta y= " + y_obje
46         else
47             print chr$(12)
48             print "Seçtiğiniz obje Nokta objesi değil!"
49         end if
50     end if
51
52 -end sub

```

Şekil 94

Şekil 46 harita üzerinde tıklanan bir nokta objesinin koordinatlarını veren Mapbasic program kodunu göstermektedir. Oluşturulan bir *ToolButton* yardımıyla harita üzerinde objelerin üzerine tıklanması sağlanmaktadır. Dikkat edilmesi gereken detay üzerine Mouse yardımıyla tıklanan obje seçili hale gelmemektedir ama objenin tüm bilgilerine ulaşılmaktadır. Yapılan işlem aşamaları:

- 1) Satır 6 ve satır 13 arası: Program ilk çalıştığında Main yordamı çalışacaktır. Main yordamında tanımlanmış olan *ToolButton* MapInfo penceresinde görünecektir. Kullanıcının Mapinfo ekranında görünen buton ile harita ekranına tıklamasıyla *nokta_koord_yaz* yordamı çalışacaktır.
- 2) Satır 22 ve Satır 23: Tıklanan yerin koordinatları alınmıştır. Koordinatlar *Commandinfo()* fonksiyonu ve fonksiyonunun *CMD_INFO_X* ve *CMD_INFO_Y* önceden tanımlanmış olan parametreleriyle elde edilmektedir.
- 3) Satır 25: Tıklanan yerde kaç adet obje olduğunu sorgulamak için *Searchpoint()* fonksiyonu kullanılıyor. Fonksiyonun ilk parametresi *Frontwindow()* fonksiyonu işlem esnasındaki aktif pencereyi ifade etmektedir. *Searcpoint()* fonksiyonunun ikinci ve üçüncü parametreleri aktif ekranda tıklanan yerin koordinatlarıdır.
- 4) Satır 27: Eğer 1 adet obje seçildiyse, seçilen objenin bulunduğu harita (tablo) adı *SearchInfo()* fonksiyonu ve ikinci parametresi *Search_Info_Table* sabit değeriyle alınır, ilgili harita içinde ki kayıt sırasını *SearchInfo()* fonksiyonu ve ikinci parametresinde *Search_Info_Row* sabit parametresiyle alınır.
- 5) Satır 34: Harita (tablo) katmanı adı ve ilgili kayıt sayısı ile grafik objeye ve ona ait belgelere ulaşılır. Bu işlem için *Fetch* kullanılır. *Fetch Rec ... From* kayda ulaşmak için kullanılır.
Fetch Rec (tablodaki kayıt sırası) From (Tablo adı)
- 6) Satır 37 ile Satır 40 arası: Ulaşılan kayıttaki grafik obje *object* değişken tipindeki bir değişkene atılır (satır 40)
- 7) Satır 40 ile Satır 45 arası: eğer obje tipi nokta grafik obje tipinde ise (satır 41) *ObjectGraphy()* fonksiyonu ile objenin X ve Y değerleri elde edilip ekrana yazılır.



Şekil 95

Grafik Objenin Harita Düzlemine Eklenmesi ve Objenin Kayıt Altına Alınması

Bir önceki uygulamalarda kullanıcıya yönelik Button (düğme) ve Dialog Form yapısı konuları anlatıldı. Yapılan örneklerde ise mesleki hesaplama konuları ve obje bilgilerinin sorgulamasına yönelik işlemler anlatıldı. Konu başlığı altında grafik objelerin hem haritaya tersim edilmesi (çizilmesi) hem de objelerin kayıt altına alınması işlemleri anlatılacak.

Haritaya obje ekleme işleminin yapılabilmesi için grafik objelerin çizim araçları içinde tanımlanmış olması gerekir. ifade 13 bir düğme içine çizim aracı tanımı da eklenmesini temsil eder. Kırmızı ile yazılmış olan kısımda çizilecek grafik tipi belirtilmelidir.

ifade 13

Create Buttonpad
Toolbutton
Drawmode [ÇİZİLECEK GRAFİK OBJE TİPİ]

Şekil 96 “MapBasic.Def” tanım dosyası içinde çizim obje tipleri için tanımlanmış grafik objelerin ifadeleri bulunmakta. Dosya incelendiğinde daire (circle), elips (ellipse), dikdörtgen (rect), line (çizgi), nokta (point), alan – çokgen (polygon) ve çokludoğru (polyline) objelerinin çizim araç tanımlarıdır.

```

-----
' Defines for different DrawModes for the custom tool.
-----
define DM_CUSTOM_CIRCLE      30
define DM_CUSTOM_ELLIPSE     31
define DM_CUSTOM_RECT        32
define DM_CUSTOM_LINE        33
define DM_CUSTOM_POINT       34
define DM_CUSTOM_POLYGON     35
define DM_CUSTOM_POLYLINE    36

```

Şekil 96

Objeye ekleme işlemi iki ayrı yöntem ile yapılabilir. Yöntemlerden birinde grafik objenin ek bilgisi olmadan sadece harita penceresine eklenmesidir. Diğer yöntemde objeye ait ek bilgilerin de kayıt altına alınarak haritaya eklenmesidir.

Grafik Objenin Ek Bilgi Olmadan Haritaya Eklenmesi

Kullanıcıdan Alınan Koordinat Değerleriyle Nokta Objesi Ekleme

Şekil 103, Şekil 104 ve Şekil 105 kullanıcının girdiği koordinat değerleriyle tabakaya nokta ekleme işlemine dair kod satırlarının olduğu resimlerdir. Programda,

- Kullanıcının harita ekranına tıklamasıyla ekrana bir form açılması,
- Açılan formda haritaya eklenecek noktanın ad, Y ve X değerlerinin girilmesi,
- Form üzerindeki ekle düğmesine tıklandığında tıklanan tabakaya nokta grafik objesinin eklenmesi istenmektedir.

Create Point Komutlarıyla Haritaya Nokta Ekleme

Program kodunda istenilenlere istinaden, program çalıştığında çizim nokta ekleme işleminin yapılacağı bir düğme gelmesi sağlanmıştır. Düğmenin program açıldığında gelmesi için düğme tasarım kodu *Main* yordamı içinde yazılmıştır (Şekil 103 6. ile 11. satırlar arasında). Düğme olarak *toolbutton* tipi düğme tercih edilmiş, düğmeye çizim modu (*Drawmode*) tipi olarak nokta tipi eklenmiştir. Düğmeye basıldığında, *diyalog_ekle* yordamı çağırılmaktadır.

Tasarlanan düğme ile ekrana tıklandığında, *diyalog_ekle* yordamı (Şekil 103 ve Şekil 104 12. ve 51. satırları arasında) çağırılmaktadır. *diyalog_ekle* yordamı içinde, kullanıcının nokta eklemek için girmesi gereken bilgilerin girileceği form tasarımına dair kod bulunmaktadır. Kod için nokta adı, noktanın Y değeri ve noktanın X değerinin girileceği metin kutuları bulunmaktadır.

Kullanıcı, formda tasarlanmış alanlara gerekli nokta bilgilerini girdikten sonra, formda tasarlanmış Nokta *Ekle* başlıklı düğmesine bastığında, *nokta_ekle* yordamı çağırılmaktadır. *nokta_ekle* yordamı (Şekil 104 53. ve 66. satırları arasında), kullanıcının veri girişini kontrol

edip, harita penceresinde *toolbutton* ile tıklanılan tabakaya noktayı grafik obje olarak eklenmesi sağlamaktadır.

Grafik objenin eklenmeden önce kullanıcının Mouse yardımıyla tıkladığı tabakanın harita tabakası olması sorgulanmalıdır. Bu işlemin yapılması için *Layerinfo()* fonksiyonunun araçları kullanılacaktır. *Layerinfo()* fonksiyonu 3 adet parametre alır. Pencerenin harita penceresi olduğunu sorugulamak için *Frontwindow()* ve *Mapperinfo()* fonksiyonlarına da ihtiyaç vardır. Bu fonksiyonların açılmaları aşağıda verilmiştir.

frontwindow():

Açık olan pencerelerden (harita penceresi, Öznitelik tablosu, grafik penceresi,...) en öndeki pencerenin (en öndeki pencere kullanıcının işlem yaptığı pencere olduğu düşünülmekte) numarasını verir.

Mapperinfo():

Mapperinfo() fonksiyonu içine iki tane parametre alır. İlk parametre hakkında soru sorulacak pencerenin sayı numarası. Ekranda var olan pencerelerden en öndeki pencerenin (işlem yapılan pencerenin) sayı numarasını almak için frontwindow() fonksiyonu kullanılır. İkinci parametre ise pencerenin harita penceresi ise pencere hakkında bilinmek istenen bilgi açıklamasıdır. Mapperinfo() fonksiyonunu Layerinfo() fonksiyonu içinde kullanılacak ve kullanım amacı tabakanında var olan tabakalar arasında ki kaçınıcı tabaka olduğu bilgisini bulmaktır. Bu sebeple mapperinfo() fonksiyonunda ikinci parametre olarak mapper_info_layers bilgisi yazılacak.

Mapperinfo(frontwindow(), Mapper_Info_Layers)

Layerinfo():

Layerinfo() fonksiyonuyla pencerenin harita penceresi olduğunu sorgulamak için kullanımı aşağıda verilmiştir. Eğer fonksiyon geriye sıfır değeri geri döndürüyorsa pencere harita penceresidir (Şekil 103 16 satır).

Layerinfo(frontwindow(), mapperinfo(frontwindow(), mapper_info_layers), layer_info_type)

Nokta ekleme işlemine başlamadan önce harita penceresinin düzenlenebilir olması soruglanmalıdır. Bu işlemde layerinfo() fonksiyonu ile yapılacaktır. Bu işlemin yapılabilmesi için Layer_info_editable özelliği sorgulanacaktır. Bu sorgunun sonucu True (doğru) yada False (yanlış) olarak geriye döner. Fonksiyonun kullanımı aşağıda verilmiştir (Şekil 103 18. satır).

Layerinfo(frontwindow(), mapperinfo(frontwindow(), mapper_info_layers), layer_info_editable)

Nokta objesi eklendikten sonra harita penceresinde limit bul işlemi yapılmıştır (Şekil 105 80. satır).

```

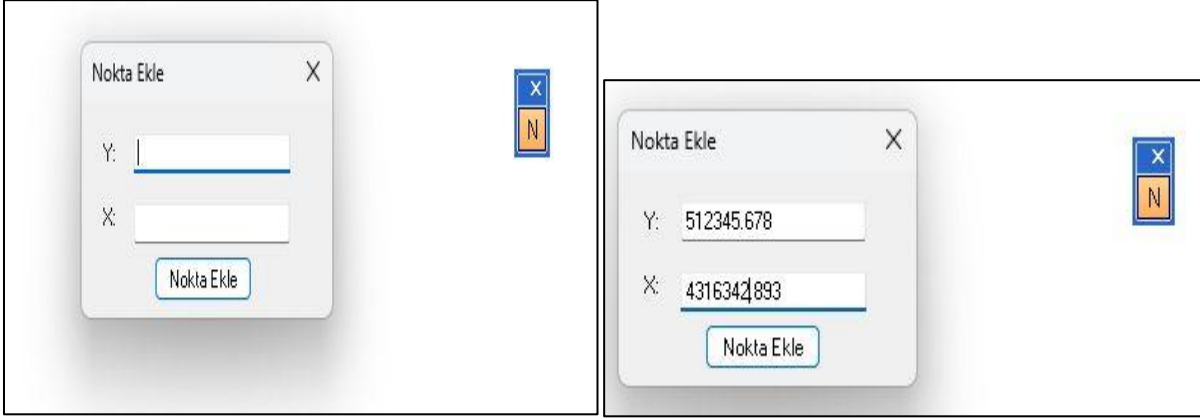
1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub nokta_dialog_ekle
5 declare sub nokta_ekle
6 Sub Main
7     create buttonpad "obje Ekle" as
8         toolbutton calling nokta_dialog_ekle
9         icon MI_ICON_LETTERS_N
10        helpmsg "Nokta objesi ekle\nNokta objesi ekle"
11 End Sub
12 sub nokta_dialog_ekle
13     if layerinfo(frontwindow(),mapperinfo(frontwindow()),Mapper_info_layers),Layer_info_type)=0 then 'harita penceresi olup olmadığı sorgulanacak
14     if layerinfo(frontwindow(),mapperinfo(frontwindow()),mapper_info_layers),layer_info_editable) then 'harita penceresinin düzenlenebilir olup olmadığı sorgulanacak
15         Dialog
16         Title "Nokta Ekle"
17         Width 131 Height 66
18
19         Control StaticText
20         Position 11, 11
21         Width 12 Height 8
22         Title "Y:"
23
24         Control EditText
25         Id 1
26         Position 27, 9
27         Width 82 Height 12
28
29         Control StaticText
30         Position 11, 31
31         Width 12 Height 8
32         Title "X:"
33

```

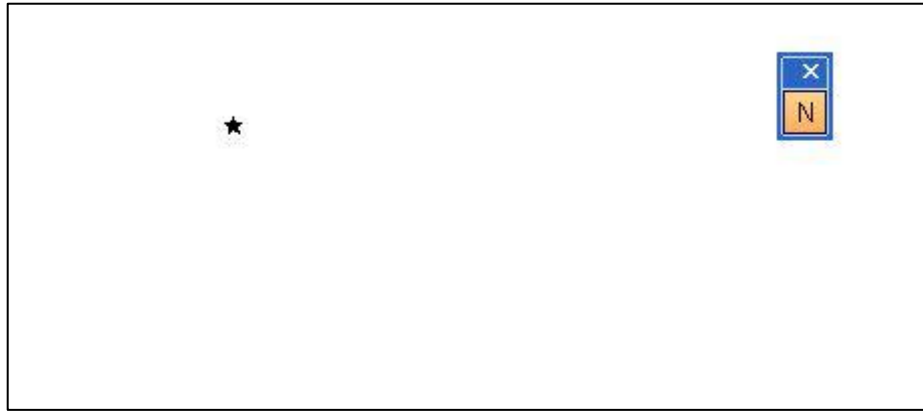
Şekil 97

```
34 Control EditText
35 Id 2
36 Position 27, 31
37 Width 82 Height 12
38
39 Control OKButton
40 Title "Nokta Ekle"
41 Position 38, 47
42 Width 51 Height 14
43 calling nokta_ekle
44 else
45 note "obje eklemek için tabakanız düzenlenebilir olmalı!"
46 end if
47
48 else
49 note "Tıkladığınız alan harita penceresi olmalı!"
50 end if
51 end sub
52
53 sub nokta_ekle
54 dim y_tut,x_tut as float
55 set coordsys window frontwindow()
56 onerror goto hatavar
57 y_tut=readcontrolvalue(1)'sağa-doğuya artan Easting
58 x_tut=readcontrolvalue(2)'yukarı-kuzaye artan Northing
59 hatavar:
60 if err()=854 then
61 note "veri girişini boş bırakmamalısınız ve koordinat değerlerini düzgün girmelisiniz"
62 elseif err()=0 then
63 create point(y_tut,x_tut)
64 set map window frontwindow() zoom entire
65 end if
66 end sub
```

Şekil 98



Şekil 99



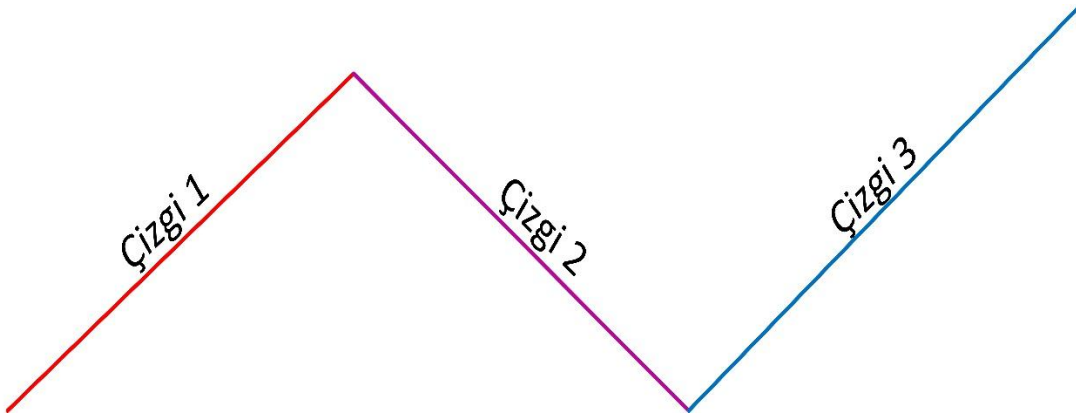
Şekil 100

Kullanıcının Çizdiği Çizgi Objenin Tabakaya Eklenmesi

Nokta grafik objesinin haritaya eklenmesi işlemlerinde iki ayrı yöntem gösterilmişti. Birinci yöntemde öznitelik verisi olmadan grafik objenin eklenmesi işlemi. İkinci yöntemde ise öznitelik bilgilerini de ekleyebilmek için Structure Query Language dili kullanılmıştı. Çizgi objesini de haritaya eklerken her iki yöntemin de kullanımı gösterilecek.

Create Line() Komutlarıyla Çizgi Grafik Objesinin Eklenmesi

Çizgi objesi kaldırım, yol ekseni gibi objelerin çiziminde kullanılır. Şekil 101 çizgi objesinin temsili görüntüsüdür. Peşi sıra çizilmiş bile olsa her bir çizgi objesi tekildir. Yol projelerinde her bir güzergâh tek bir çizgiden oluşur. Yolun tamamı birçok güzergâhtan oluşabilir. Çizgi objesi başlangıç ve bitiş noktalarından oluşur.



Şekil 101

Şekil 102 çizgi grafik objesini create line() parametreleriyle eklenmesine bir örnektir. Çizgi objesi başlangıç ve bitiş objelerinden oluşur. Commandinfo() fonksiyonu yardımıyla çizginin birinci ve ikinci noktalarını alarak grafik obje çizilmiştir.

```

1  Include "MapBasic.def"
2  include "icons.def"
3  Declare Sub Main
4  declare sub çizgi_ekle
5  Sub Main()
6
7      create buttonpad "Objeye Ekleme" as
8          toolbutton calling çizgi_ekle
9          drawmode DM_CUSTOM_LINE
10         icon MI_ICON_LETTERS_C
11         helpmsg "Çizgi ekleme işlemi yapar\nÇizgi ekleme işlemi yapar"
12 End Sub
13 sub çizgi_ekle
14     dim x1,y1,x2,y2 as float
15     set coordsys window frontwindow()
16     x1=commandinfo(CMD_INFO_X)
17     y1=commandinfo(CMD_INFO_Y)
18     x2=commandinfo(CMD_INFO_X2)
19     y2=commandinfo(CMD_INFO_Y2)
20     create line(y1,x1)(y2,x2)
21     set map window frontwindow() zoom entire
22 end sub
23

```

Şekil 102

Grafik Objenin Ek Bilgileriyle Beraber Harita Penceresine Eklenmesi**Structure Query Language (SQL) Komutlarıyla Haritaya Nokta Ekleme**

Nokta grafik objesi Create Point komutlarıyla harita düzlemine eklendiğinde dikkat edilmesi gereken noktaya ait bilgilerin (noktanın adı, noktanın nirengi – poligon – detay noktası olduğu,...) tabloya eklenemediğidir. Create point komutları sadece nokta grafik objesinin haritaya eklenmesini sağlamaktadır.

Noktaya ait bilgilerin de tabloya eklenmesi isteniyorsa Coğrafi Bilgi Sistemleri yazılımlarının tablo özellikleri kullanılmalıdır. Coğrafi bilgi sistemi yazılımları hem grafik objeyi harita üzerinde saklar hem de grafik objeyi niteleyen, grafik objeyi bir diğer grafik objeden ayıran bilgilerin de saklanmasını sağlar. Bu niteleyici bilgiler öznitelik olarak ifade edilir.

Coğrafi bilgi sistemleri yazılımlarında bir tabaka oluşturulmaya başlandığında kayıt altına alınacak grafik objelerin öznitelik bilgilerinin girilmesini sağlayacak pencerede açar. Objeyi niteleyen bilgilerin sayısı sınırlı değildir. Kayıt altına alınacak bilgi sayısı kullanıcının yapacağı uygulamaya ve kullanıcının kayıt altına almak istediği bilgilere bağlıdır. Ayrıca bu bilgilerin sayı, alfabetik, tarih,... gibi farklı veri tipinde olması da sağlanabilir.

Hem grafik objenin eklenmesi hem de grafik objeye ait öznitelik bilgilerinin eklenmesi için Structure Query Language (SQL) dili kullanılır. Türkçe karşılığı Yapısal Sorgulama Dili

olarak ifade edilir. SQL dili kullanılarak grafik objenin ve öznelik verilerinin eklenmesi, verilerin güncellenmesi, verilerin silinmesi, hem grafik hem de öznelik veriler arasında sorgulama yapılabilmesi sağlanır.

SQL dili ile kullanılarak oluşturulan tabakanın haritasına veya öznelik tablosuna eklemek işlemi için sabit komutlar kullanılmaktadır.

ifade 14

Insert Into tablo_adi(öznelik1, öznelik2, ..., obj) values(veri1, veri2, ..., objoluşturmafonksiyonu)

Insert kelimesinin Türkçe karşılığı ekleme anlamındadır. Into esasında in to olarak İngilizce'de kullanılır ve Türkçe karşılığı içine anlamındadır. Into komutundan sonra tablonun adı ve parantez içinde veri eklenecek olan özneliklerin adları yazılır. Values parametresinde parantez içinde belirtilen özneliklere eklenecek veriler ve haritaya eklenecek grafik objeyi oluşturan fonksiyon eklenir.

ifade 14 kullanılarak nokta grafik objesini ekleyebilmek için tablonun adı bilinmeli ve values parametresinde grafik objeyi eklemek için kullanılacak fonksiyon bilinmelidir.

Tabaka adının alınabilmesi için *LayerInfo()* fonksiyonu kullanılmıştır (Şekil 105 78. satır). Bu işlemin yapılması için *Layer_info_name* bilgisi kullanılır (Şekil 105 78. satır).

layerinfo(frontwindow(), mapperinfo(frontwindow(), Mapper_info_layers), Layer_info_name)

Nokta grafik objesinin eklenmesi için values parametresinde *Createpoint()* fonksiyonu kullanılır. *Createpoint()* fonksiyonu *Create Point()* yapısında olduğu gibi ilk önce sağa artan koordinat değeri sonrasında yukarı artacak koordinat değeri kullanılır (Şekil 105 79. satır). Yapılan örnekte nokta grafik objesi ve nokta ad öznelik bilgisi eklenmiştir. Tablo oluşturulurken noktanın sadece ad öznelik bilgisi eklenecek şekilde düzenleme yapılmıştır.

```

1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub diyalog_ekle
5 declare sub nokta_ekle
6
7 Sub Main
8     create buttonpad "Objeekele" as
9     toolbutton calling diyalog_ekle
10    icon MI_ICON_LETTERS_N
11    Drawmode DM_CUSTOM_POINT
12    helpmsg "Nokta eklemek için ekrana tıklayın\nNokta eklemek için ekrana tıklayın"
13 End Sub
14
15 sub diyalog_ekle
16     if layerinfo(frontwindow()),mapperinfo(frontwindow()),mapper_info_layers),layer_info_type)=0 then
17
18         if layerinfo(frontwindow()),mapperinfo(frontwindow()),mapper_info_layers),layer_info_editable) then
19             Dialog
20             Title "Nokta Ekle"
21             Width 146 Height 94
22
23             Control StaticText
24             Position 9, 8
25             Width 18 Height 8
26             Title "Nno"
27
28             Control StaticText
29             Position 9, 25
30             Width 12 Height 8
31             Title "Y:"
32
33             Control StaticText
34             Position 9, 45
35             Width 12 Height 8
36             Title "X:"

```

Şekil 103

```

38 Control EditText
39 Id 1
40 Position 39, 8
41 Width 82 Height 12
42
43 Control EditText
44 Id 2
45 Position 39, 25
46 Width 82 Height 12
47
48 Control EditText
49 Id 3
50 Position 39, 43
51 Width 82 Height 12
52
53 Control Button
54 Title "Nokta Ekle"
55 Position 46, 65
56 Width 51 Height 14
57 calling nokta_ekle
58 else
59 note "Harita Penceresinin düzenleme ayarı kapalı!"
60 end if
61 else
62 note "İşlem yaptığınız pencere harita penceresi değil!"
63 end if
64 end sub
65
66 sub nokta_ekle
67 dim x_tut,y_tut as float
68 dim no_tut,tablo_adi as string
69 set coordsys window frontwindow()
70 onerror goto hatavar
71 y_tut=readcontrolvalue(1)
72 x_tut=readcontrolvalue(2)
73 no_tut=readcontrolvalue(3)

```

Şekil 104

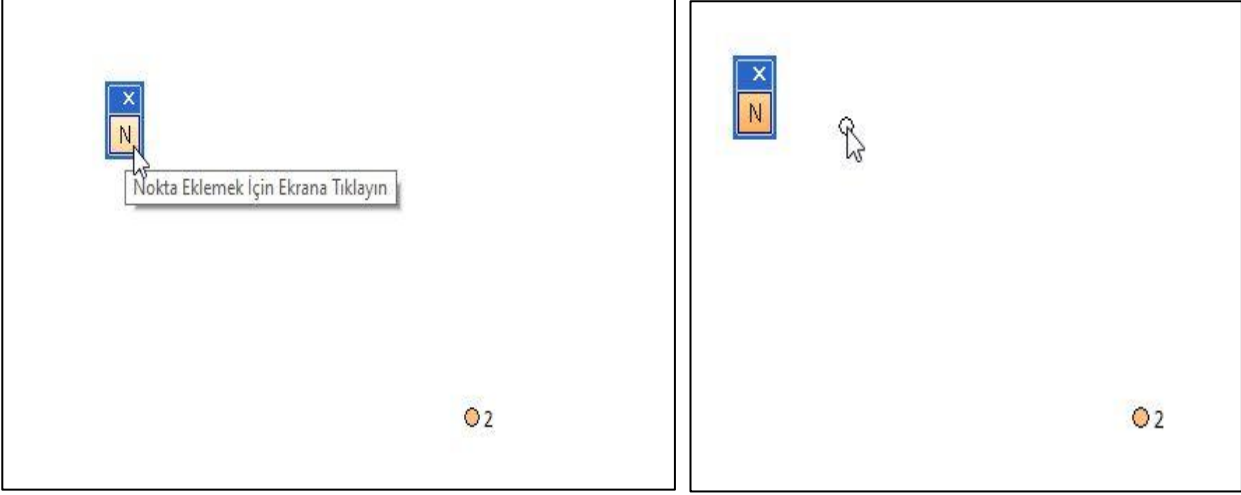
```

71 y_tut=readcontrolvalue(1)
72 x_tut=readcontrolvalue(2)
73 no_tut=readcontrolvalue(3)
74 hatavar:
75 if err()=854 then
76 note "veri girişi yapılmalıdır, veri kutularına uygun veri girilmelidir!"
77 elseif err()=0 then
78 tablo_adi=layerinfo(frontwindow(),mapperinfo(frontwindow()),Mapper_info_layers),Layer_info_name)
79 insert into tablo_adi(nokta_ad,obj) values(no_tut,createpoint(y_tut,x_tut))
80 Set Map Window FrontWindow() Zoom Entire
81 end if
82
83 end sub

```

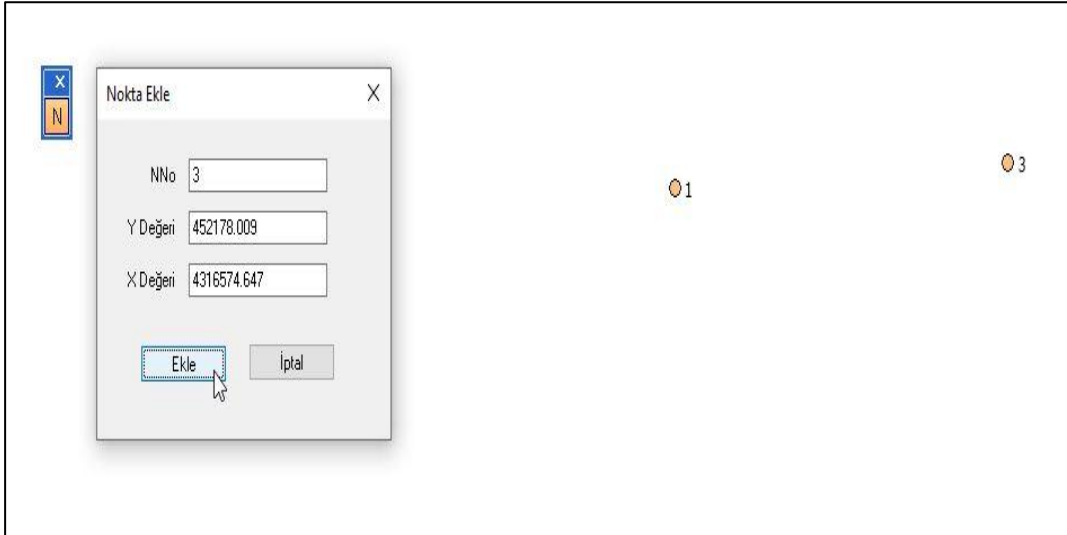
Şekil 105

Şekil 106 ve Şekil 107 program kodunun çalıştığında programın kullanımının tasviridir. Program çalıştığında ilk önce düğme ekrana gelir (Şekil 106 sol resim). Düğmenin seçilmesiyle, kullanıcıdan ekrana tıklanması gerekir. Ekranı tıkladığında, nokta eklenmesi için gerekli bilgilerin girileceği form ekrana gelir (Şekil 107). Forma gerekli bilgiler girildikten sonra Ekle düğmesine basıldığında nokta tabakaya eklenir ve harita penceresinde gözükür (Şekil 107).



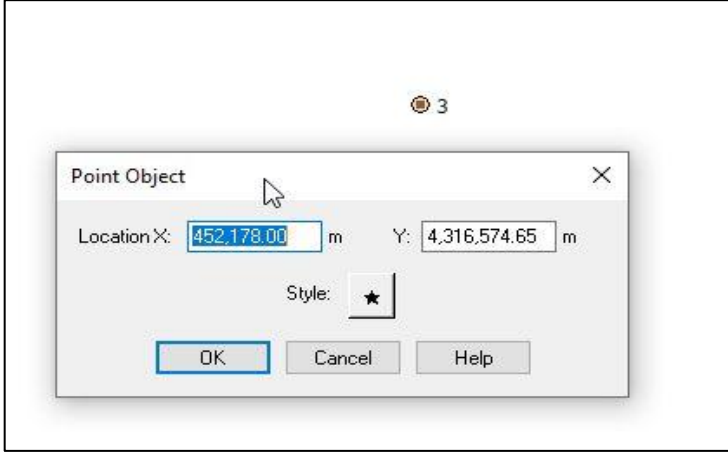
Şekil 106

Şekil 107 forma eklenen koordinat değerleriyle 3 adlı noktanın tabakaya eklenmesi ve harita penceresinde gösteriminin tasviri vardır.



Şekil 107

Şekil 108 3 adlı noktanın üzerine çift tıklanması ile açılan obje bilgisini gösteren pencerenin görüntüsüdür. Kullanıcının girdiği koordinat ile sorgu sonucu gözükür koordinat değeri aynıdır.



Şekil 108

Kullanıcının Nokta Ekleme veya Noktanın Koordinat Bilgilerinin Güncellemesini Yapan program

Şekil 109, Şekil 110, Şekil 111, Şekil 112 ve Şekil 113 resimleri kullanıcının nokta ekleme ve seçilen noktanın koordinat bilgilerinin güncellemesini sağlayan program kodunu içerirler. Programda istenen:

- Kullanıcının bir düğme kullanarak harita penceresine tıklaması,
- Eğer harita düğme aracı ile tıkladığı noktada daha önce eklenmiş bir nokta grafik objesi yok ise tabakaya nokta eklemek için form açılmasının sağlanması,
- Eğer harita penceresine düğme aracı ile tıkladığı noktada daha önceden bir nokta grafik objesi varsa, o grafik objesinin nokta koordinatlarını güncellenmesini sağlayan program kodunun oluşturulması

İstenilenlerin yapılması için ilk olarak düğme eklenmesi gerekir. Toolbutton tipinde düğme, programın çalıştırıldığında harita penceresine gelmesi için düğme kod bloğu main yordamı içine eklenmiştir (Şekil 109 10. ve 20. satırları arasında).

Dikkat edilmesi gereken, düğme ile hem nokta ekleme hem de seçilen grafik objenin koordinatlarının güncellenmesi işlemi yapılacaktır. MapInfo programı, Nectad yazılımından farklı olarak, güncelleme işleminin yapılabilmesi için ilk olarak objenin seçilmesini istemektedir. MapInfo yazılımının bu özelliği de düşünülerek program tasarlanmıştır. Eğer düğme ile harita penceresinde boş bir noktaya tıklanırsa, nokta ekleme için kullanılacak form açılması; harita penceresinde daha önceden seçilmiş olan bir objeye tasarlanan düğme ile tıklanırsa noktanın güncellenmesi için forma noktanın var olan koordinat bilgileri eklenmesi tasarlanmıştır.

İki ayrı işlemin yapılabilmesi için ilk önce tasarlanan düğme kullanılarak harita penceresi üzerinde tıklanan noktada bir obje seçimi yapılıp yapılmadığını sorgulayan *kontrol_et*

yordamı oluşturulmuştur (Şekil 109 21. ve 28. satırları arasında). Seçim işleminin sorgulanması için *SelectionInfo()* fonksiyonu kullanılmıştır. Eğer harita üzerindeki bir objenin seçim yaptığı seçilirse, güncelleme işlemi için formun ekrana getirecek *gun_form* yordamı çağrılacak; eğer yapmadığı tespit edilirse *ek_form* yordamı çağrılacak.

Güncelleme işlemlerinde kullanılacak forma, kullanıcının seçtiği nokta grafik objesine ait koordinat değerleri, doldurulacak. Kullanıcı düğme ile harita ekranında boş bir noktaya tıkladığında nokta ekleme formu ekrana gelecek.

Güncelleme işlemlerinin yapılabilmesi için *guncelle* isimli yordam çağırılmaktadır. Güncelleme işlemi için Alter ve Update SQL komutları kullanılmaktadır (Şekil 112 ve Şekil 113 resimlerinde 128. ve 144. satırlar arasında).

Nokta grafik objesinin tabakaya eklenmesi için *nokta_ek* adlı yordam oluşturulmuştur (Şekil 113 resminde 145. ile 165. satırları arasında). Ekleme işlemleri için Insert SQL komutu kullanılmıştır.

```

1 include "mapbasic.def"
2 include "icons.def"
3 declare sub main
4 declare sub kontrol_et
5 declare sub gun_form
6 declare sub ek_form
7 declare sub guncelle
8 declare sub nokta_ek
9 declare sub temizle
10 sub main
11     Print chr$(12)
12     create buttonpad "Nokta" as
13         toolbutton
14         id 1
15         calling kontrol_et
16         icon MI_ICON_SYMBOL
17         cursor MI_CURSOR_ARROW
18         Drawmode DM_CUSTOM_POINT
19         HelpMsg "Haritada Bir Noktaya Tıklayın\nHaritada Bir Noktaya Tıklayın"
20 end sub
21 sub kontrol_et
22     set coordsys window frontwindow()
23     if selectioninfo(SEL_INFO_NROWS)=1 then
24         call gun_form
25     elseif selectioninfo(SEL_INFO_NROWS)=0 then
26         call ek_form
27     end if
28 end sub
29 sub gun_form
30
31 dim x_tik,y_tik as float
32     x_tik=commandinfo(CMD_INFO_X)
33     y_tik=commandinfo(CMD_INFO_Y)
34     Dialog

```

Şekil 109

```
35 Title "Nokta Ekle"  
36 Width 173 Height 116  
37  
38 Control StaticText  
39 Position 31, 16  
40 Width 20 Height 8  
41 Title "NNo"  
42  
43 Control StaticText  
44 Position 18, 35  
45 Width 33 Height 8  
46 Title "Y Deęeri"  
47  
48 Control StaticText  
49 Position 18, 54  
50 Width 33 Height 8  
51 Title "X Deęeri"  
52  
53 Control EditText  
54 Id 2  
55 Position 54, 14  
56 Width 82 Height 12  
57  
58 Control EditText  
59 Id 3  
60 Position 54, 33  
61 Width 82 Height 12  
62 value str$(x_tik)  
63 Control EditText  
64 Id 4  
65 Position 54, 52  
66 Width 82 Height 12  
67 value str$(y_tik)  
68 Control Button
```

Şekil 110

```
69         Title "Ekle"  
70         Id 5  
71         Position 25, 82  
72         Width 51 Height 14  
73         calling guncelle  
74     Control CancelButton  
75         Title "İptal"  
76         Id 6  
77         Position 89, 81  
78         Width 51 Height 14  
79  
80 end sub  
81 sub ek_form  
82     Dialog  
83     Title "Nokta Ekle"  
84     Width 173 Height 116  
85  
86     Control StaticText  
87     Position 31, 16  
88     Width 20 Height 8  
89     Title "NNo"  
90  
91     Control StaticText  
92     Position 18, 35  
93     Width 33 Height 8  
94     Title "Y Deęeri"  
95  
96     Control StaticText  
97     Position 18, 54  
98     Width 33 Height 8  
99     Title "X Deęeri"  
100  
101     Control EditText  
102     Id 2
```

Şekil 111

```
103         Position 54, 14
104         Width 82 Height 12
105
106     Control EditText
107         Id 3
108         Position 54, 33
109         Width 82 Height 12
110
111     Control EditText
112         Id 4
113         Position 54, 52
114         Width 82 Height 12
115
116     Control Button
117         Title "Ekle"
118         Id 5
119         Position 25, 82
120         Width 51 Height 14
121         calling nokta_ek
122     Control CancelButton
123         Title "İptal"
124         Id 6
125         Position 89, 81
126         Width 51 Height 14
127 end sub
128 sub guncelle
129     dim xgun,ygun as float
130     dim objem as object
131     dim tablo_ad as string
132     fetch first from selection
133     ygun=readcontrolvalue(3)
134     xgun=readcontrolvalue(4)
135     objem=selection.obj
136     alter object objem Geography OBJ_GEO_POINTX,ygun
```

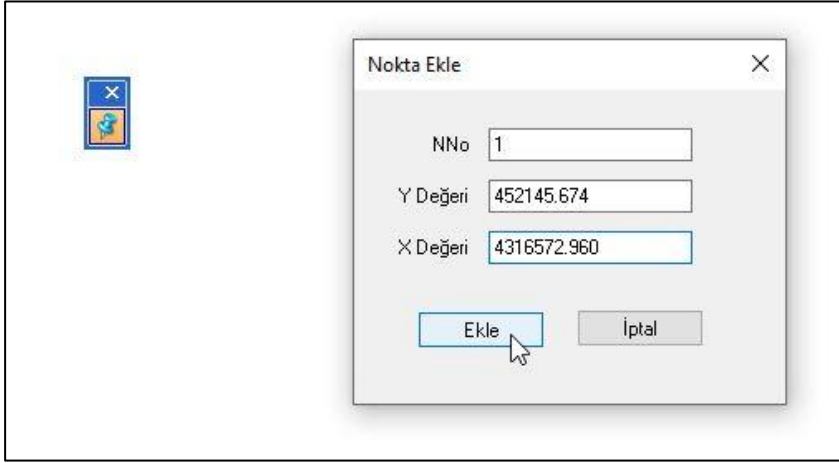
Şekil 112

```
137 alter object objem Geography OBJ_GEO_POINTY,xgun
138 tablo_ad=selectioninfo(SEL_INFO_SELNAME)
139 update tablo_ad
140     Set obj=objem
141 print chr$(12)
142 print "Nokta Güncellendi"
143 call temizle
144 end sub
145 sub nokta_ek
146     dim x,y as float
147     dim pencere_id as integer
148     dim tablo_ad,no_ad as string
149     dim obje as object
150 set coordsys window frontwindow()
151 pencere_id=frontwindow()
152 if windowinfo(pencere_id,WIN_INFO_TYPE)=WIN_MAPPER then
153     'nokta ekleme
154     no_ad=readcontrolvalue(2)
155     y=readcontrolvalue(3)
156     x=readcontrolvalue(4)
157     tablo_ad=layerinfo(pencere_id,mapperinfo(pencere_id,MAPPER_INFO_LAYERS),LAYER_INFO_NAME)
158     insert into tablo_ad(Nokta_ad,obj) values(no_ad,createpoint(y,x))
159     print chr$(12)
160     print "Nokta "+tablo_ad+" isimli tabakaya eklendi"
161 else
162     note "İşlemin Yapılması için Harita Penceresine Tıklamalısınız!"
163 end if
164 call temizle
165 end sub
166 sub temizle
167     Alter control 2 value ""
168     Alter control 3 value ""
169     Alter control 4 value ""
170 End Sub
```

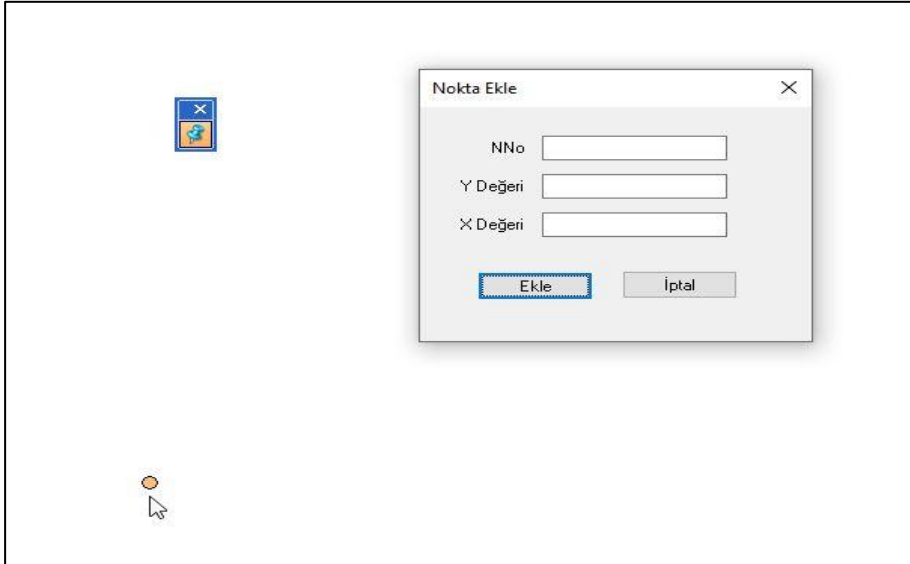
Şekil 113



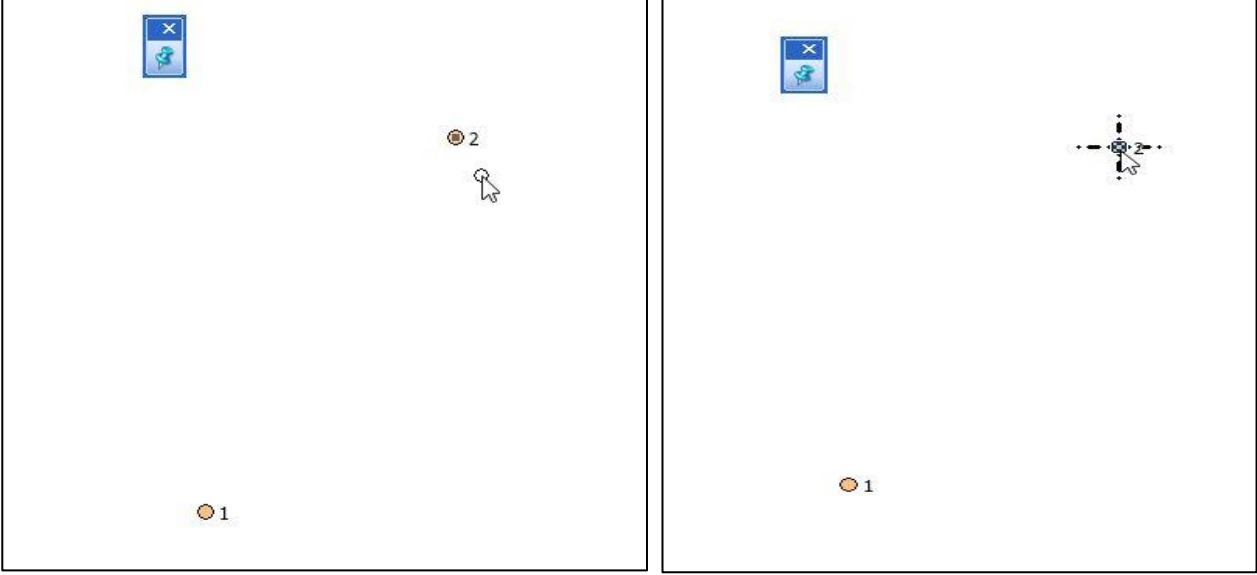
Şekil 114



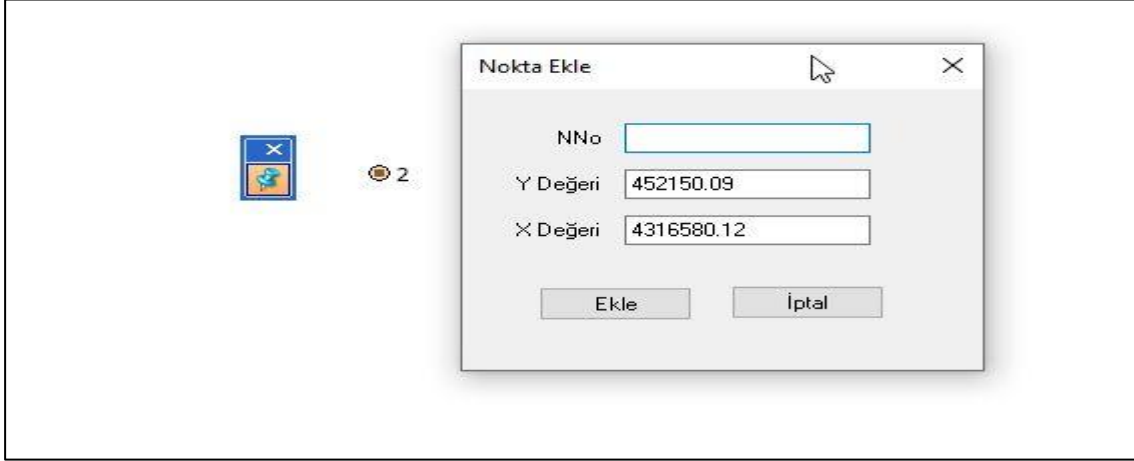
Şekil 115



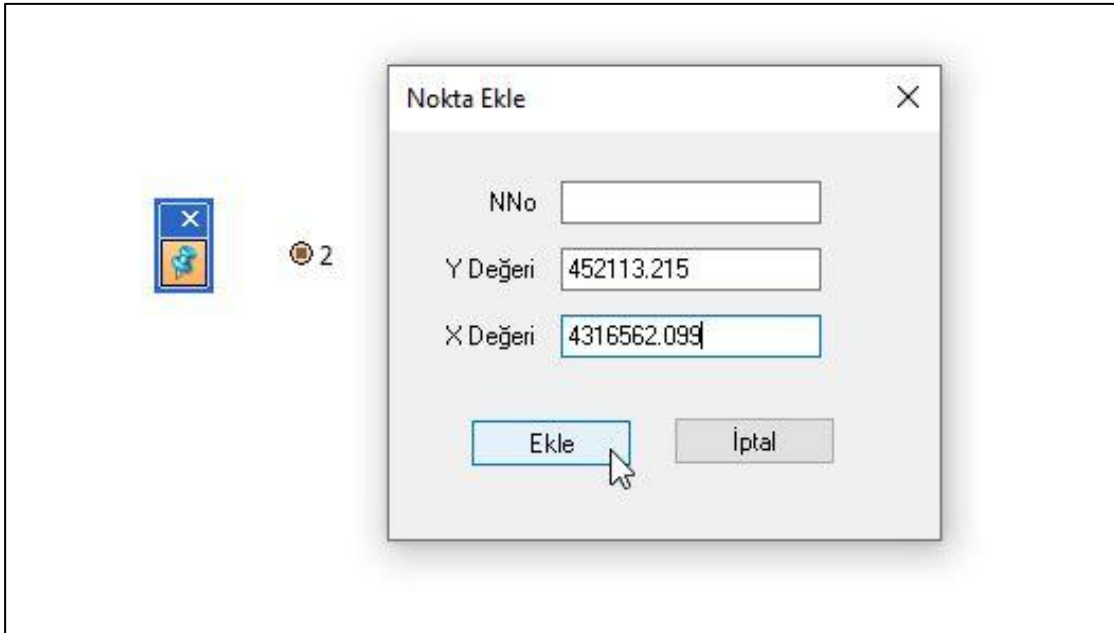
Şekil 116



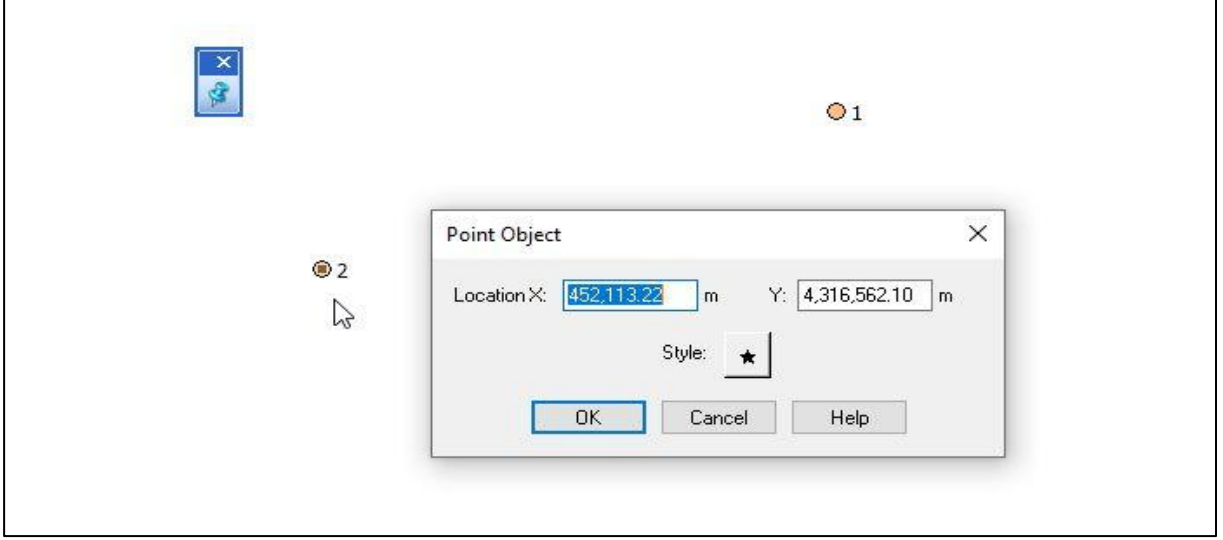
Şekil 117



Şekil 118



Şekil 119



Structure Query Language (SQL) Komutlarıyla Çizgi Objesini Ekleme

Çizgi grafik nesnesini özellikleriyle haritaya eklemek için ,Nokta grafik nesnesini SQL cümlecikleriyle eklemek işleminin benzeri uygulanacak.

insert into tablo_ad(cizgi_ad, obj) values(cizgi_adi, createline(x1, y1, x2, y2))

İşlemlerin yapılmasında farklı olarak çizgi adının alınmasında dialog parametreleri içinde into ifadesi kullanılmıştır (Şekil 121 55. satır). Şekil 122 ve Şekil 123 program kodunun çalıştırılması sonucu elde edilen görüntülerdir. Çizgi tabakasının etiketi açık olduğu için çizgi nesnesinin özellik bilgisi olan adı, grafik nesnenin üzerinde görülmektedir (Şekil 123).

```

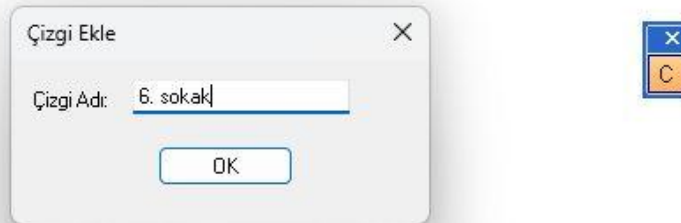
1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub cizgi_ekle
5 declare sub cizgi_ad_al
6 dim cizgi_adi as string
7 Sub Main()
8     cizgi_adi=""
9     create buttonpad "Çizgi Ekle" as
10         toolbutton calling cizgi_ekle
11         icon MI_ICON_LETTERS_C
12         Drawmode DM_CUSTOM_LINE
13         helpmsg "Çizgi Ekleme aracı\nÇizgi Ekleme aracı"
14 End Sub
15 sub cizgi_ekle
16     dim x1,y1,x2,y2 as float
17     dim tablo_ad as string
18     set coordsys window frontwindow()
19     if layerinfo(frontwindow(),mapperinfo(frontwindow(),mapper_info_layers),Layer_info_type)=0 then
20         if layerinfo(frontwindow(),mapperinfo(frontwindow(),mapper_info_layers),layer_info_editable) then
21             tablo_ad=layerinfo(frontwindow(),mapperinfo(frontwindow(),mapper_info_layers),layer_info_name)
22             x1=commandinfo(CMD_INFO_X)
23             y1=commandinfo(CMD_INFO_Y)
24             x2=commandinfo(CMD_INFO_X2)
25             y2=commandinfo(CMD_INFO_Y2)
26             call cizgi_ad_al
27             insert into tablo_ad(cizgi_ad,obj) values(cizgi_adi,createline(x1,y1,x2,y2))
28             cizgi_adi=""
29         else
30             note "çizgi eklemek için tabakanın düzenlenebilir olması gerekir!"
31         end if
32     else
33         note "çizgi ekleme işlemi için harita penceresini kullanmalısınız!"
34     end if
35 end sub

```

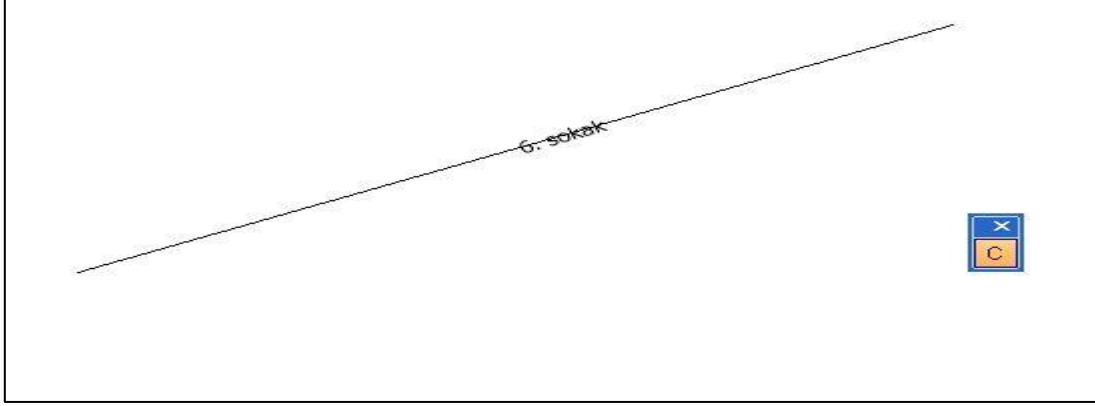
Şekil 120

```
37 sub cizgi_ad_al
38   Dialog
39   Title "Çizgi Ekle"
40   Width 158 Height 58
41
42   Control OKButton
43   Title "OK"
44   Position 55, 30
45   Width 51 Height 14
46
47   Control StaticText
48   Position 8, 10
49   Width 34 Height 8
50   Title "Çizgi Adı:"
51
52   Control EditText
53   Position 46, 7
54   Width 82 Height 12
55   into cizgi_adi
56 end sub
57
```

Şekil 121



Şekil 122



Şekil 123

Structure Query Language (SQL) Komutlarıyla Haritaya Çoklu Doğru Ekleme

```

1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub cokludogru_ekle
5 declare sub dogru_ad_al
6 Sub Main
7   create buttonpad "obje Ekle" as
8     toolbutton calling dogru_ad_al
9     Icon MI_ICON_LETTERS_N
10    Drawmode DM_CUSTOM_POLYLINE
11    helpmsg "Çoklu doğru ekler\nÇoklu doğru ekler"
12 End Sub
13 sub dogru_ad_al
14   if layerinfo(frontwindow(),mapperinfo(frontwindow(),MAPPER_INFO_LAYERS),LAYER_INFO_TYPE)=0 then
15     if layerinfo(frontwindow(),mapperinfo(frontwindow(),MAPPER_INFO_LAYERS),LAYER_INFO_EDITABLE) then
16       Dialog
17         Title "Çoklu Doğru Ekle"
18         Width 175 Height 51
19         Control StaticText
20           Position 9, 8
21           Width 34 Height 8
22           Title "Çizgi Adı:"
23         Control EditText
24           Id 1
25           Position 47, 7
26           Width 82 Height 12

```

```

27     Control Button
28     Title "Ekle"
29     Position 55, 30
30     Width 51 Height 14
31     calling cokoludogru_ekle
32     else
33     note "harita düzenlenebilir olmalı!"
34     end if
35     else
36     note "Objeye harita penceresine eklenebilir!"
37     end if
38 end sub
39 sub cokoludogru_ekle
40     dim opline as object
41     dim adi as string
42     onerror goto hatavar
43     adi=readcontrolvalue(1)
44     hatavar:
45     if err()=854 then
46     note "veri girişinde hata var!"
47     elseif err()=0 then
48     set coordsys window frontwindow()
49     opline=commandinfo(CMD_INFO_CUSTOM_OBJ)
50     insert into plineekleme(ad,obj) values(adi,opline)
51     set map window frontwindow() zoom entire
52     end if
53 End Sub

```

Kullanıcının Çizdiği Çokgen (Alan) Grafik Objesinin Tabakaya Eklenmesi

Şekil 124 ve Şekil 125 alan grafik objesinin harita penceresine eklenmesine dair kod bloğuna ait resimlerdir. Bir önceki örnekte nokta objesini tabakaya eklerken:

insert into tabaka_adi(saha,obj) values(saha_değeri,creatpoint(x,y))

kod satırı kullanıldı. Kod satırı incelendiğinde, values() içindeki parametrelerden, creatpoint() parametresi yardımıyla nokta grafik objesi eklenmiş oldu.

Çoklu doğru (polyline) veya çokgen (polygon) grafik objelerinin eklenmesi için creatpoint() gibi bir fonksiyon yok. Tanımlanan düğmelere çoklu doğru veya çokgen grafik objelerinin çizim modu atanabiliyor. Çokgen grafik objesinin çizimi için düğme objesine DM_CUSTOM_POLYGON çizim modu atanıyor. Çoklu doğru grafik objesini çizimi için

düğme objesine DM_CUSTOM_POLYLINE çizim modu atanıyor. Bu çizim modları ile çokgen veya çoklu doğru objelerinin çizimi yapılabiliyor. Objelerin tablolara kayıtlarının yapılabilmesi için Commandinfo() fonksiyonu kullanılıyor. Commandinfo() en son yapılan olaylar (obje çizimi, seçim yapılması, pencere açılması, Mouse ile tıklanması gibi olaylar) hakkında geriye veri döndürür. Çoklu doğru veya çokgen objelerinin tabloya aktarılması için commandinfo() fonksiyonun en son yapılan işlemi geriye döndürmesi kullanılacak. commandinfo() fonksiyonu sayesinde en son yapılan alan veya çoklu doğru çizimi ile oluşan grafik obje, object tipinde bir değişkene aktarılabilir. Bu sayede çoklu doğru veya çokgen grafik objeleri tabloya kayıt olabilecekler.

Şekil 124 ve Şekil 125 örneğini incelendiğinde benzer kod satırı, Şekil 125 53.satırda bulunmaktadır. ifade 15 commadinfo() çalışma prensibini göstermektedir. obje_a object veri tipinde bir değişkendir. Commandinfo() fonksiyonuna eklenen *CMD_INFO_CUSTOM_OBJ* parametresi sayesinde en son çizilen alan veya çoklu doğru grafik obje, object veri tipinde bir değişkene aktarılabilir. Şekil 125 53.satırda objenin kayıt işlemi de yapılabiliyor.

ifade 15

obje_a = commandinfo(CMD_INFO_CUSTOM_OBJ)

Bu örnek te diğer örneklerdeki düğmelerden farklı olarak, tanımlama yaparken Show ve Fixed komutları kullanılmıştır. Fixed komutu sayesinde eklenen düğme araç çubuğuna sabitlenmiştir (Şekil 124 15.satır).

```
1 include "mapbasic.def"
2 include "icons.def"
3 declare sub main
4 declare sub alan_ad_al
5 declare sub alan_ekle
6 sub main
7     create buttonpad "Alan" as
8         toolbar
9             drawmode DM_CUSTOM_POLYGON
10            icon MI_ICON_POLYGON
11            Helpmsg "Alan Çizim Aracı\nAlan Çizim Aracı"
12            calling alan_ad_al
13            cursor MI_CURSOR_LRG_CROSSHAIR
14        show
15        fixed
16    end sub
17    sub alan_ad_al
18        Dialog
19        Title "Çokgen Adı"
20        Width 146 Height 63
21
22    Control StaticText
23        Position 9, 15
24        Width 31 Height 8
25        Title "Alan Adı"
26
27    Control EditText
28        Id 1
29        Position 44, 13
30        Width 82 Height 12
```

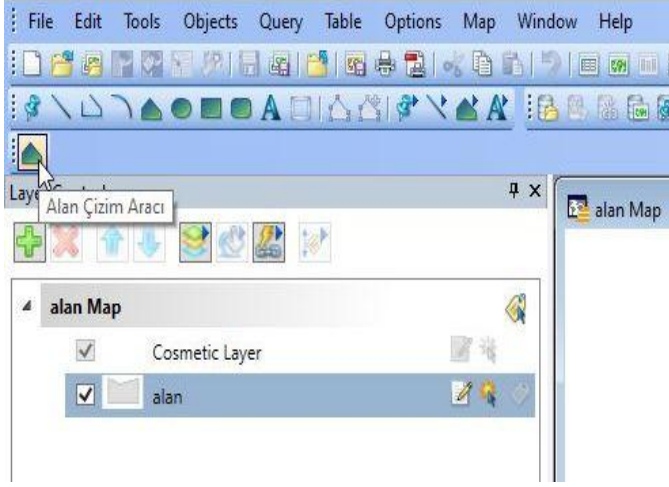
Şekil 124

Bir

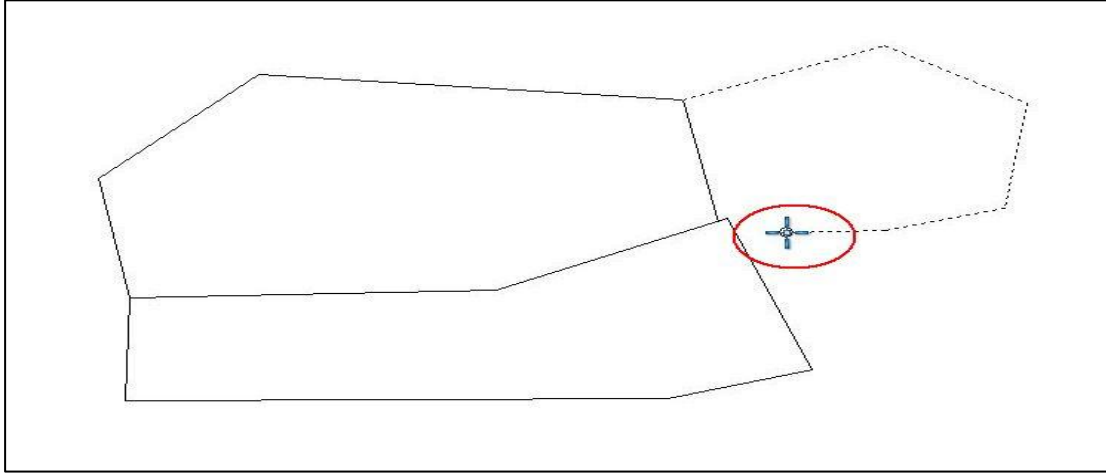
```
32 Control okButton
33     Title "Ekle"
34     Position 18, 36
35     Width 51 Height 14
36     calling alan_ekle
37 Control CancelButton
38     Title "Çık"
39     Position 73, 36
40     Width 51 Height 14
41 end sub
42 sub alan_ekle
43     dim pencere_id as integer
44     dim obje_a as object
45     dim tabaka_ad, alan_adi as string
46     set coordsys window frontwindow()
47     pencere_id=frontwindow()
48     if mapperinfo(pencere_id,Mapper_info_edit_layer)>=0 then
49         if readcontrolvalue(1)<>"" then
50             alan_adi=readcontrolvalue(1)
51             tabaka_ad=layerinfo(pencere_id,mapperinfo(pencere_id,Mapper_info_layers),LAYER_INFO_NAME)
52             obje_a=commandinfo(CMD_INFO_CUSTOM_OBJ)
53             insert into tabaka_ad(Alan_ad,obj) values(alan_adi,obje_a)
54         else
55             Note "Alan adı girmelisiniz!"
56         end if
57     else
58         Note "İşlem Yapılacak Tabaka Düzenlenebilir olmalı!"
59     end if
60 end sub
```

Şekil 125

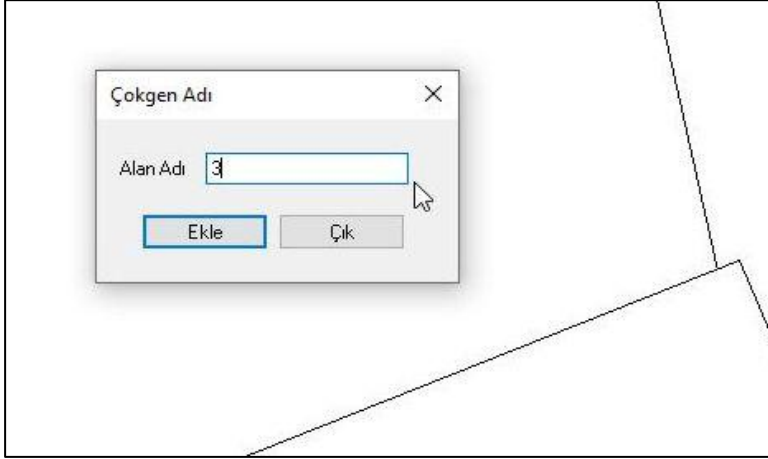
Alan grafik objesi, harita penceresine eklenirken kullanıcıdan alan grafik objesinin adı bilgisi de istenmiştir. Alan grafik objesinin kaydedildiği tabakada bu bilgi veri olarak kayıt edilmektedir.



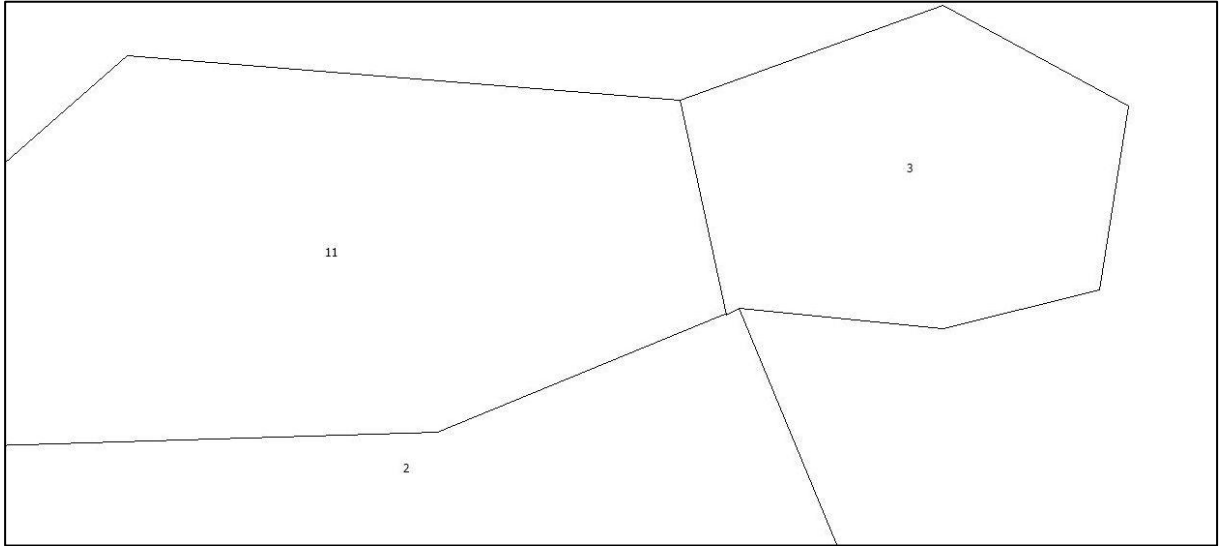
Şekil 126



Şekil 127



Şekil 128



Şekil 129

Birden Fazla Aracı Tek Bir ButtonPad içinde toplama

```

birdenfazlabuton.mb
1  Include "MapBasic.def"
2  include "icons.def"
3  Declare Sub Main
4  declare sub obje_algila_cevapla
5
6  Sub Main()
7      create buttonpad "Sorgu" as
8
9          toolbutton
10             calling obje_algila_cevapla id 1
11             helpmsg "Poligon Alanı Bul\nPoligon Alanı Bul"
12             icon MI_ICON_POLYGON
13         toolbutton
14             calling obje_algila_cevapla id 2
15             helpmsg "Çizgi Uzunluğu Bul\nÇizgi Uzunluğu Bul"
16             icon MI_ICON_LINE
17         toolbutton
18             calling obje_algila_cevapla id 3
19             helpmsg "Nokta Koordinatı Bul\nNokta Koordinatı Bul"
20             icon MI_ICON_CROSSHAIR
21         title "SORGU"
22         width 8
23     End Sub
24
25 sub obje_algila_cevapla
26     dim hangi_buton as integer
27     dim x_tik , y_tik, x_obje, y_obje as float
28     dim tablo_ad as alias
29     dim obje_tip , pencere_id, kayit_deger , i as integer
30     dim obje as object
31     set coordsys window frontwindow()
32     pencere_id=frontwindow()
33     if windowinfo(pencere_id,WIN_INFO_TYPE)<>WIN_MAPPER then
34         print chr$(12)
35         print "Bu araç harita penceresinde olmayan bir pencerede kullanılamaz"
36     else
37         x_tik=commandinfo(CMD_INFO_X)
38         y_tik=commandinfo(CMD_INFO_Y)
39         i=searchpoint(frontwindow(),x_tik,y_tik)
40         if i=0 then
41             print chr$(12)
42             Print "Bir obje seçmelisiniz"

```

Şekil 130

birdenfazlabuton.mb

```

43     elseif i=1 then
44         tablo_ad=searchinfo(i,SEARCH_INFO_TABLE)
45         kayıt_deger=searchinfo(i,SEARCH_INFO_ROW)
46         fetch rec kayıt_deger from tablo_ad
47         tablo_ad=tablo_ad+".obj"
48         obje=tablo_ad
49         obje_tip=objectinfo(obje,OBJ_INFO_TYPE)
50
51         hangi_buton=commandinfo(CMD_INFO_TOOLBTN) 'seçilen butonun id değerini geri döndüren fonksiyon
52
53         if hangi_buton=1 then
54             if obje_tip=OBJ_TYPE_REGION then
55                 print chr$(12)
56                 print "Alan: "+ round( CartesianArea(obje, "sq m"),0.01)
57             else
58                 print chr$(12)
59                 print "Seçtiğiniz Obje Alan objesi değil"
60             end if
61
62             elseif hangi_buton=2 then
63                 if obje_tip=OBJ_TYPE_LINE or obje_tip=OBJ_TYPE_PLINE then
64                     print chr$(12)
65                     print "uzunluk: " +Round(CartesianObjectLen( obje, "m" ),0.001)
66                 else
67                     print chr$(12)
68                     print "Seçtiğiniz Obje Doğru veya Çoklu Doğru objesi değil!"
69                 end if
70
71             elseif hangi_buton=3 then
72                 if obje_tip=OBJ_TYPE_POINT then
73                     x_obje = ObjectGeography(obje, OBJ_GEO_POINTX)
74                     y_obje = ObjectGeography(obje, OBJ_GEO_POINTY)
75                     print chr$(12)
76                     print "Nokta x= " + x_obje +" Nokta y= " + y_obje
77                 else
78                     print chr$(12)
79                     print "Seçtiğiniz obje Nokta objesi değil!"
80                 end if
81             end if 'hangi buton olduğunu sorugulayan if bitimi
82         end if 'eğer bir adet obje seçilmişse işlem yapılması
83     end if 'eğer harita penceresine tıklandıysa işlem yapılması
84
85 end sub

```

Şekil 131

Şekil 130 ve Şekil 131 Harita üzerinde Tıklanan objelerin koordinat, uzunluk ve alan bilgilerini ekrana yazdıran *Toolbutton* kod örneklerinin tek bir ButtonPad üzerinde toplayarak

işlemleri daha etkin bir hale gelmesini sağlamıştır. Program kodu incelendiğinde satır 49'a kadar program kodu önceki örneklere göre aynıdır. Satır 51 hangi butona basıldığını belirleyen koddur. *CommadInfo(CMD_INFO_TOOLBTN)* seçilen butonun id değerini döndürür. 53. Satır ile 83. Satır arası seçilen buton türüne göre işlem yapan program kodunu olduğu bloktur.

Menü Eklenmesi

Mapinfo programı çalıştığında üstteki araç çubuklarında menüler bulunmaktadır (Şekil 132). Her menü altında birden fazla alt menü olabilir. Her menü belirli bir amaç için toplanmış alt menüleri gruplandırmak için kullanılır. Alt menüler, daha önce hazırlanmaya çalışılan buton araçları gibi yordam çağırabilirler.



Şekil 132

Menü Eklenmesi İşlemi

Menülerin eklenmesi aynı düğme objelerinin eklenmesi gibi main yordamı içinde yapılır. Bu sayede program çalıştığında düğme araç çubuğuna eklenmiş olur. Aşağıda menü eklenmesi için bir örnek bulunmaktadır.

Create menu “Menü Adı” as

“Birinci Alt Menü adı”

Helpmsg “Menü üzerinde Mouse ile durulduğunda çıkacak mesaj”

Calling yordam_cagirilmesi,

“ikinci Alt menü adı”

Helpmsg “Menü üzerinde Mouse ile durulduğunda çıkacak mesaj”

Calling yordam_cagirilmesi

Alter menu bar add “Menü Adı”

Yukarıdaki MapBasic program kodu incelendiğinde, düğme eklenmesine benzer bir şekilde *Create Menu* ifadesiyle menü oluşturulması başlanmış olur. “Menü Adı” ifadesi, menüye verilecek olan addır. Örnekte iki adet alt menü eklenmiştir. Birinci menü ifadesini bitimi *Calling yordam_cagirilmesi*, ifadesindeki , (virgül) ile sağlanmıştır. İkinci menü için

program kodu yazıldıktan sonra *Alter menu bar add* "Menü Adı" ifadesiyle, menu bar olarak ifade edilen normal menülerin olduğu araç çubuğuna oluşturulan menü eklenir.

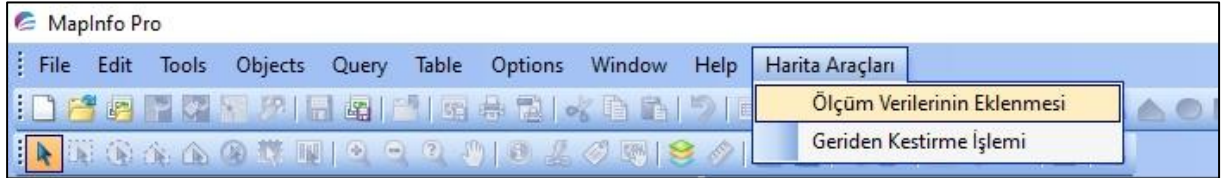
Şekil 133 program kodu çalıştırıldığında Şekil 134 görünen menü ve alt menüleri oluşur.

```

1 include "mapbasic.def"
2 include "icons.def"
3 declare sub main
4 declare sub deneme1
5 declare sub deneme2
6 sub main
7     create menu "Harita Araçları" as
8         "Ölçüm Verilerinin Eklenmesi"
9             Helpmsg "Yatay açı ve yatay mesafe değerleri ile nokta ekleme"
10            calling deneme,
11            "Geriden Kestirme İşlemi"
12            calling deneme
13 alter menu bar add "Harita Araçları"
14 end sub
15
16 sub deneme1
17     print "Harita Araçları menüsü çalıştı"
18 end sub
19
20 sub deneme2
21     print "Geriden Kestirme İşlemi menüsü çalıştı"
22 end sub
23

```

Şekil 133



Şekil 134

Elektronik Takeometre ile Yapılan Ölçüm verilerinin Eklenmesi

```
1 Include "MapBasic.def"
2 include "icons.def"
3 Declare Sub Main
4 declare sub nokta_dugme
5 declare sub nokta_sec
6 declare sub semt_hesapla
7 declare sub ilk_aci_diyalog
8 declare sub ilk_aci_kontrol
9 declare sub nokta_ymya_diyalog
10 declare sub nokta_ekle
11 declare sub temizle
12 declare function ysemt_hesapla(byval yadeger as float)as float
13 dim nokta_sayi as integer
14 dim xtut1,ytut1,xtut2,ytut2,semt_aci,pi,ilk_aci as float
15
16 Sub Main()
17     nokta_sayi=0
18     semt_aci=0
19     ilk_aci=0
20     pi=3.141592654
21     xtut1=0
22     ytut1=0
23     xtut2=0
24     ytut2=0
25     create menu "Harita Araçları" as
26         "Yatay Açığı - Yatay Mesafe"
27         helpmsg "Yatay Açığı ve Yatay mesafe ile koordinat hesabı"
28         calling nokta_dugme
29     alter menu bar add "Harita Araçları"
30
```

```
33 sub nokta_dugme
34   print chr$(12)
35   print "birinci noktayı seç"
36   create buttonpad "Noktaseç" as
37     toolbutton calling nokta_sec
38     drawmode DM_CUSTOM_POINT
39     ICON MI_ICON_LETTERS_N
40 end sub
41
42 sub nokta_sec
43   nokta_sayi=nokta_sayi+1
44   set coordsys window frontwindow()
45   if nokta_sayi=1 then
46     xtut1=commandinfo(CMD_INFO_Y)
47     ytut1=commandinfo(CMD_INFO_X)
48     print chr$(12)
49     print "ikinci noktayı seç"
50   elseif nokta_sayi=2 then
51     xtut2=commandinfo(CMD_INFO_Y)
52     ytut2=commandinfo(CMD_INFO_X)
53   end if
54   if xtut2>0 and ytut2>0 then
55     call semt_hesapla
56     call ilk_aci_diyalog
57   end if
58 end sub
```



```
59 sub semt_hesapla
60   if (ytut2-ytut1)>0 and (xtut2-xtut1)>0 then
61     semt_aci=atn((ytut2-ytut1)/(xtut2-xtut1))*200/pi
62   elseif (ytut2-ytut1)>0 and (xtut2-xtut1)<0 then
63     semt_aci=200 - atn((ytut2-ytut1)/(xtut2-xtut1)*(-1))*200/pi
64   elseif (ytut2-ytut1)<0 and (xtut2-xtut1)<0 then
65     semt_aci=200 + atn((ytut2-ytut1)/(xtut2-xtut1))*200/pi
66   else
67     semt_aci=400 - atn((ytut2-ytut1)/(xtut2-xtut1)*(-1))*200/pi
68   end if
69 end sub
70 sub ilk_aci_diyalog
71   Dialog
72     Title "İlk Açısı"
73     Width 199 Height 71
74     Control StaticText
75       Id 1
76       Position 61, 6
77       Width 15 Height 8
78       Title "-----"
79     Control StaticText
80       Position 9, 23
81       Width 61 Height 8
82       Title "İlk Açısı Değeri Gir:"
83     Control EditText
84       id 2
85       Position 74, 23
86       Width 82 Height 12
87     Control OKButton
88       Title "İlk Açısı Ekle"
```

```
89     Position 74, 39
90     Width 51 Height 14
91     calling ilk_aci_kontrol
92 end sub
93 sub ilk_aci_kontrol
94     dim dene as float
95     onerror goto hatavar
96     dene=readcontrolvalue(2)
97     hatavar:
98     if err()=854 then
99         note "ilk açı değerinde hata var!"
100    elseif err()=0 then
101        ilk_aci=dene
102        call nokta_ymya_diyalog
103    end if
104
105 End Sub
106 sub nokta_ymya_diyalog
107     Dialog
108         Title "Takeometri Değerleri"
109         Width 177 Height 86
110         Control StaticText
111             Position 39, 6
112             Width 20 Height 8
113             Title "Nno:"
114         Control EditText
115             Id 3
116             Position 65, 6
117             Width 82 Height 12
118         Control StaticText
```

```

119     Position 22, 28
120     Width 37 Height 8
121     Title "Yatay Açı:"
122     Control EditText
123     Id 4
124     Position 65, 26
125     Width 82 Height 12
126     Control StaticText
127     Position 10, 48
128     Width 51 Height 8
129     Title "Yatay Mesafe:"
130     Control EditText
131     Id 5
132     Position 65, 46
133     Width 82 Height 12
134     Control Button
135     Title "Nokta Ekle"
136     Position 65, 62
137     Width 51 Height 14
138     calling nokta_ekle
139
140 end sub
141

```

```

142 sub nokta_ekle
143     dim ad_tut,tablo_ad as string
144     dim ya_tut,ym_tut,semt_tut,x_bul,y_bul as float
145     set coordsys window frontwindow()
146     onerror goto hatavar
147     ad_tut=readcontrolvalue(3)
148     ya_tut=readcontrolvalue(4)
149     ym_tut=readcontrolvalue(5)
150     hatavar:
151     if err()=854 then
152         note "veri girişinde hata var!"
153     elseif err()=0 then
154         tablo_ad=layerinfo(frontwindow(),Mapperinfo(frontwindow()),MAPPER_INFO_LAYERS),LAYER_INFO_NAME)
155         semt_tut=ysemt_hesapla(ya_tut)
156         x_bul=xtut1 + ym_tut*cos(semt_tut*pi/200)
157         y_bul=ytut1 + ym_tut*sin(semt_tut*pi/200)
158         insert into tablo_ad(ad,obj) values(ad_tut,createpoint(y_bul,x_bul))
159         set map window frontwindow() zoom entire
160         call temizle
161     end if
162 End Sub
163

```

```

164 function ysem_t_hesapla(byval yadeger as float)as float
165     dim kir_aci as float
166
167     if (yadeger>ilk_aci) then
168         kir_aci=yadeger-ilk_aci
169     elseif ilk_aci>yadeger then
170         kir_aci=400+yadeger-ilk_aci
171     end if
172     if (sem_t_aci+kir_aci)<200 then
173         ysem_t_hesapla=sem_t_aci+kir_aci+200
174     elseif (sem_t_aci+kir_aci)<600 and (sem_t_aci+kir_aci)>200 then
175         ysem_t_hesapla=sem_t_aci+kir_aci-200
176     elseif (sem_t_aci+kir_aci)>600 then
177         ysem_t_hesapla=sem_t_aci+kir_aci-600
178     elseif (sem_t_aci+kir_aci)=200 then
179         ysem_t_hesapla=0
180     elseif (sem_t_aci+kir_aci)=400 then
181         ysem_t_hesapla=200
182     end if
183 end function
184 sub temizle
185     alter control 3 value ""
186     alter control 4 value ""
187     alter control 5 value ""
188 end sub

```

Geriden Kestirme Hesabı

Yatay Mesafe Değeri Kullanılarak Geriden Kestirme Hesabı

Hata! Başvuru kaynağı bulunamadı. P.8 noktasının koordinatlarının geriden kestirme yöntemi ile hesaplanmasına dair ölçümleri temsil etmektedir. Geriden kestirme hesaplama yöntemi ile P.8 noktasının koordinatları bulunacağı için ölçümler P.8 noktasından yapılmaktadır. P.3 ve P.4 noktaları ölçümde kullanılacak koordinatları bilinen noktalardır. Hesaplama yönteminde yatay mesafe değerleri kullanılacaktır. P.8 noktasının koordinatlarını, yatay mesafe değerleri kullanarak geriden kestirme yöntemi ile hesaplamak için, P.8 noktasından hem P.3 noktasına hem de P.4 noktasına yatay mesafe değeri okuması yapılmıştır. P.3 ve P.4 numaralı noktaların koordinatları ve P.8 noktasından yapılan ölçüm verileri aşağıdaki tablolarda verilmiştir.

NNo	Y	X
P.3	560068.086	4358232.750

P.4	560127.051	4358230.673
-----	------------	-------------

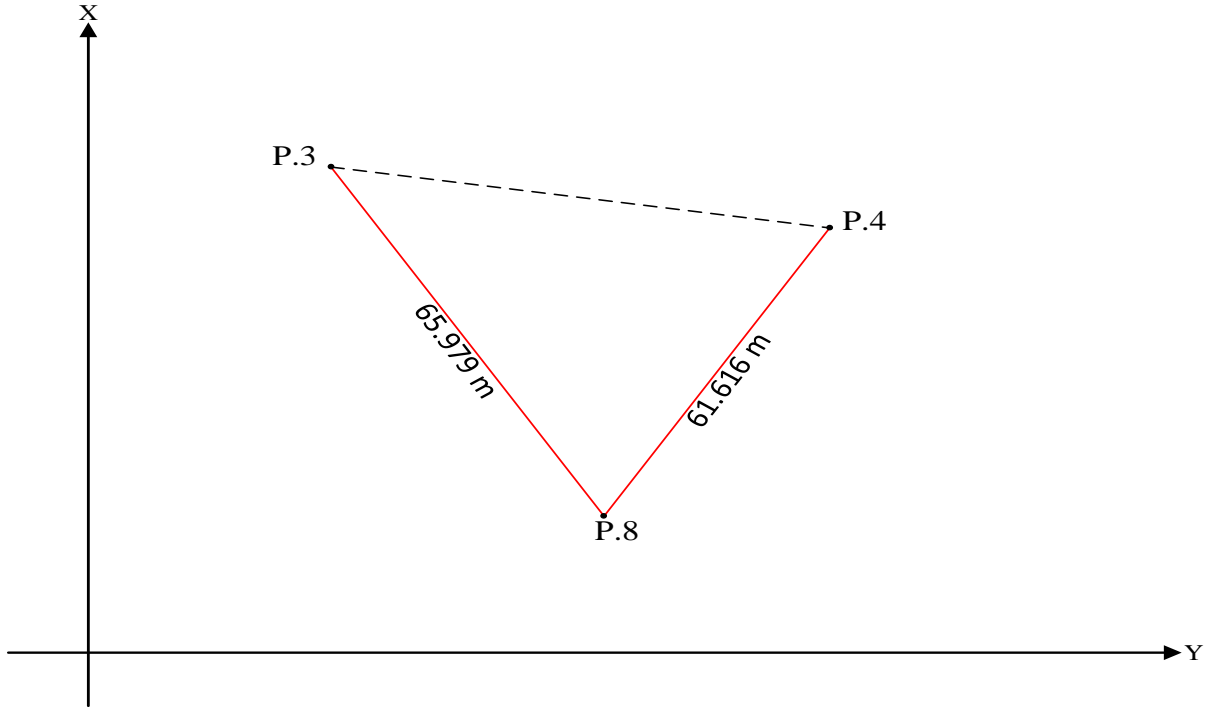
DN	BN	YM
P.8	P.3	65.979 m

	P.4	61.616 m
--	-----	----------

Şekil 136 incelendiğin P.8 noktasından yapılan ölçüm ile bir üçgen oluşturulur. Oluşan bu üçgende P.8 noktasından P.3 ve P.4 noktalarına yapılan ölçümlerden yatay mesafe değerleri elde edilecektir. Bu değerler ve genel üçgen çözüm formülleri kullanılarak P.8 noktasının koordinatları bulunması hedeflenmektedir.



Şekil 135



Şekil 136

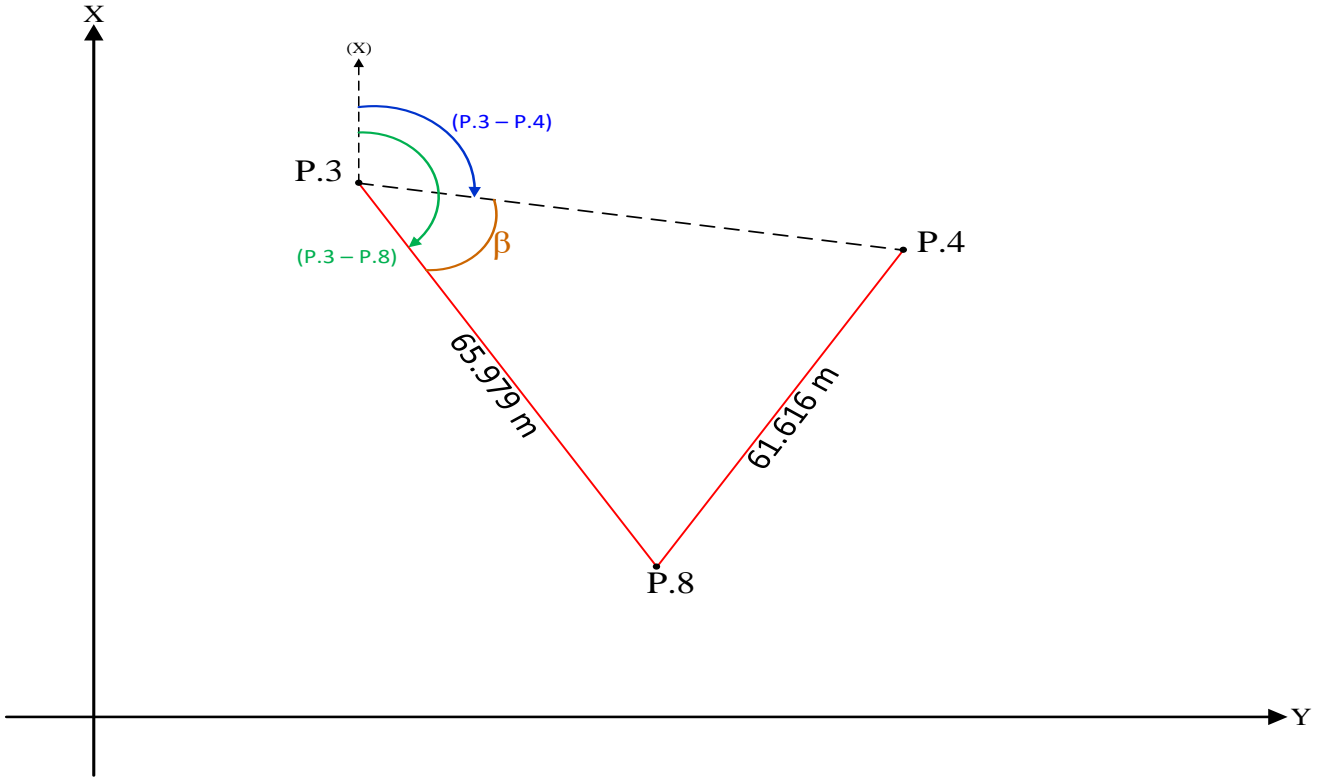


Geriden Kestirme hesabında yapılacak ölçümler koordinatı bulunacak noktadan yapılır.

P.8 numaralı noktanın koordinatları hem P.3 noktasından hesaplanacak hem de P.4 noktasından hesaplanacaktır. Her iki noktadan da yapılan hesaplamaların sonucunda $Y_{P.8}$ ve $X_{P.8}$ koordinat değerleri aynı çıkmalıdır.

P.8 numaralı noktanın koordinatlarının P.3 noktasından hesaplanması için $(P.3 - P.8)$ semt açısına ihtiyaç vardır. $(P.3 - P.8)$ semt açısının bulunması içinde $(P.3 - P.4)$ ve β açılarının elde edilmesi gerekmektedir.

$$(P.3 - P.8) = (P.3 - P.4) + \beta$$



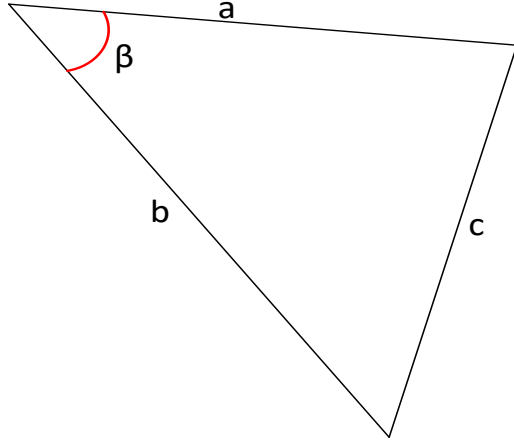
Şekil 137

$(P.3 - P.4)$ semt açısı P.3 ve P.4 noktasının koordinatlarından hesaplanabilir. β açısının bulunması için var olan üçgen kenar uzunlukları ve kosinüs teoremi kullanılmalıdır.

Şekil 138 kosinüs teoremi ve üçgen kenar uzunlukları kullanılarak üçgen içindeki bir açının nasıl hesaplanacağını temsil eder.

a, b, c uzunlukları biliniyor

β bulunmak isteniyor



$$c^2 = a^2 + b^2 - 2 * a * b * \cos(\beta)$$

$$2 * a * b * \cos(\beta) = a^2 + b^2 - c^2$$

$$\cos(\beta) = \frac{a^2 + b^2 - c^2}{2 * a * b}$$

$$\beta = \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2 * a * b}\right)$$

Şekil 138

Çözüm:

1) $(P.3 - P.4)$ semt açısının hesaplanması

$$\left. \begin{array}{l} Y_{P.4} - Y_{P.3} \text{ pozitif} \\ X_{P.4} - X_{P.3} \text{ negatif} \end{array} \right\} 2. \text{ Bölge}$$

$$(P.3 - P.4) = 200^g - \tan^{-1}((Y_{P.4} - Y_{P.3}) \div (X_{P.4} - X_{P.3}) * (-1)) = 102.2415^g$$

2) $\overline{P.3 - P.4}$ mesafesinin hesaplanması

$$\overline{P.3 - P.4} = \sqrt{((Y_{P.4} - Y_{P.3})^2 + (X_{P.4} - X_{P.3})^2)} = 59.002 \text{ m}$$

3) β açısının hesaplanması

$$\beta = (\overline{P.3 - P.4}^2 + 65.979^2 - 61.616^2) \div (2 * \overline{P.3 - P.4} * 65.979 \text{ m})$$

$$\beta = 65.2887^g$$

4) $(P.3 - P.8)$ semt açısının hesaplanması

$$(P.3 - P.8) = (P.3 - P.4) + \beta = 167.5303^g$$

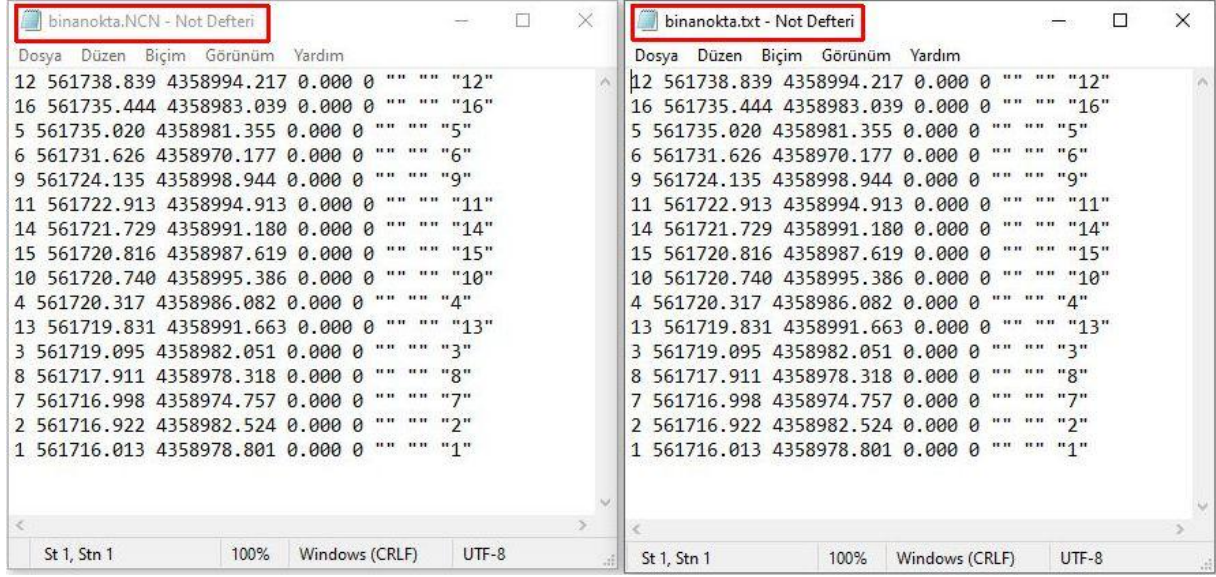
5) P.8 numaralı noktanın yatay düzlem koordinatlarının hesaplanması

$$Y_{P.8} = Y_{P.3} + 65.979 \text{ m} * \sin(P.3 - P.8) = 560100.297 \text{ m}$$

$$X_{P.8} = X_{P.3} + 65.979 \text{ m} * \cos(P.3 - P.8) = 4358175.168 \text{ m}$$

Metin Dosyalarının Okunması

Metin dosyaları text (string) veri tipinde verilerin saklandığı dosya tipleridir. Not defteri yazılımlarıyla açılabilen ve düzenlenebilen dosya tipidir. CAD yazılımlarında detay verilerinin koordinatlarının saklandığı, GNSS sinyal alıcısı veya elektronik takeometre cihazlarında tutulan nokta koordinatlarının dışarıya aktarılması veya cihaz içerisine aktarılmasında metin dosyaları kullanılır. Şekil 139 nokta dosyalarının saklandığı NCN ve txt uzantılı metin dosyaları örneğidir.



Şekil 139

Şekil 139 bina objelerinin detaylarına ait nokta koordinatlarının tutulduğu Netcad nokta dosyası (Şekil 139 sol resim- NCN uzantılı dosya) ve aynı dosyanın metin dosyası (Şekil 139 sağ resim - TXT uzantılı dosya) gösterimini içerir. MapInfo yazılımı NCN uzantılı dosyaları açamaz. Bu sebepten dolayı, NCN uzantılı dosyanın uzantısı txt olacak şekilde değiştirilmiştir. Dosyanın uzantısı dışında dosya üzerinde bir değişiklik yapılmamıştır. NCN uzantısının txt uzantısına dönüştürülmesiyle Mapbasic yazılımında nokta dosyası içindeki bilgilere erişilebilecektir.

Şekil 140 txt uzantılı nokta dosyasındaki her bir satırın sırayla okunup, MapInfo yazılımında oluşturulmuş *nokta* adlı tabakaya (Şekil 141) nokta grafik objesinin eklenmesini sağlayan program kodudur.

Şekil 140 satır 18'de tasarlanan düğmeye (Şekil 142) basıldığında kullanıcının metin dosyasını seçmesini sağlayacak bir pencere açılacaktır (Şekil 143). Açılan pencerede kırmızı kutu içine alınan dosya türü kısmı, satır 18'de yapılan tanım ile sağlanır. Şekil 143 kullanıcının txt uzantılı dosyayı ilgili dizinden seçmesini sağlayan penceredir. Dosya seçimi yapıldıktan sonra dosya adı değişkene aktarılır (satır 18). Satır 19 dosyanın açılmasını sağlar ve dosyaya 1 numaralı dosya tanımını yapar. 20.satır ile 31.satırlar arasında do while – loop döngüsü ile tüm dosya satırları ayrı ayrı okunur. Do while – loop döngüsündeki sınıma *Not EOF(1)* ile sağlanmıştır. EOF(), end of file – dosya sonu fonksiyonu ile kontrol dosyanın son satırına ulaşıldığını kontrol eder. Döngü sınamasında kullanılan *Not EOF(1)*, okunan yeni satırın 1 numaralı dosyanın son satırı olup olmadığını kontrol eder, eğer dosyanın son satırı değilse yeni satır okumaya devam edilmesini sağlar. Diğer bir izahı dosya sonu olmadığı sürece okumaya devam edilmesidir. Her yeni okunan satır, *satır* adlı değişkene aktarılır ve satır içinde ki nokta adı, y ve x değerleri sırasıyla okunup

ilgili değişkenlere aktarılır. Değerlere ulaşabilmek için *instr()* ve *mid\$()* fonksiyonları kullanılmıştır.

Satır içindeki Nokta adı, y ve x değerleri arasında bir tab boşluğu kadar boşluk vardır.

$bos1 = instr(1, satir, " ") \rightarrow$ kod satırında (satır 24), dosyadan okunan her bir satırın, 1. karakterinden itibaren tab boşluğu aranır. Bulunan boşluk kaçınıcı karakterde ise, *instr()* fonksiyonu bu değeri geri döndürür. Kod satırında bulunan değer *bos1* adlı değişkene aktarılıyor.

İlk boşluk karakterinin satır içinde kaçınıcı karakter olduğu bulunduktan sonra Satır içindeki değerlerin her birine ulaşabilmek için *mid\$()* metin fonksiyonu kullanılmıştır. Fonksiyon birinci parametresinde satır bilgisini alır. İkinci parametresinde metin ifade içinden çekeceği karakterin başlangıç numarasını alır. Üçüncü parametresindeyse, başlangıç karakterinden itibaren kaç karakter alacağı belirlenir. Örneğin dosya içindeki ilk satır:

12 561738.839 4358994.217 0.000 0 "" "" "12"

\rightarrow boşluk başlangıçtan itibaren 3. Karakter.

mid(satir, 1, (bos1 - 1)) \rightarrow$ kod satırında, dosyanın ilk satırı *satir* adlı değişkendedir. fonksiyonun ilk parametresi *satir* değişkenidir. Kod içinde *satir* değişkeninde 1. Karakterden itibaren (*bos1-1*) kadar (yani 2 karakter) değer çekilmesini sağlar. Sonuç 12 olacaktır.

mid(satir, (bos1+1), 10) \rightarrow$ *satir* değişkeni içindeki metinden (*bos1+1*) karakterden (yani 4.karakter) başlayarak 10 karakter çekilmesini sağlar. Sonuç *561738.839* olacaktır.

mid(satir, (bos1+12), 11) \rightarrow$ *satir* değişkeni içindeki metinden (*bos1+12*) karakterden (yani 15.karakter) başlayarak 11 karakter çekilmesini sağlar. Sonuç *4358994.217* olacaktır.

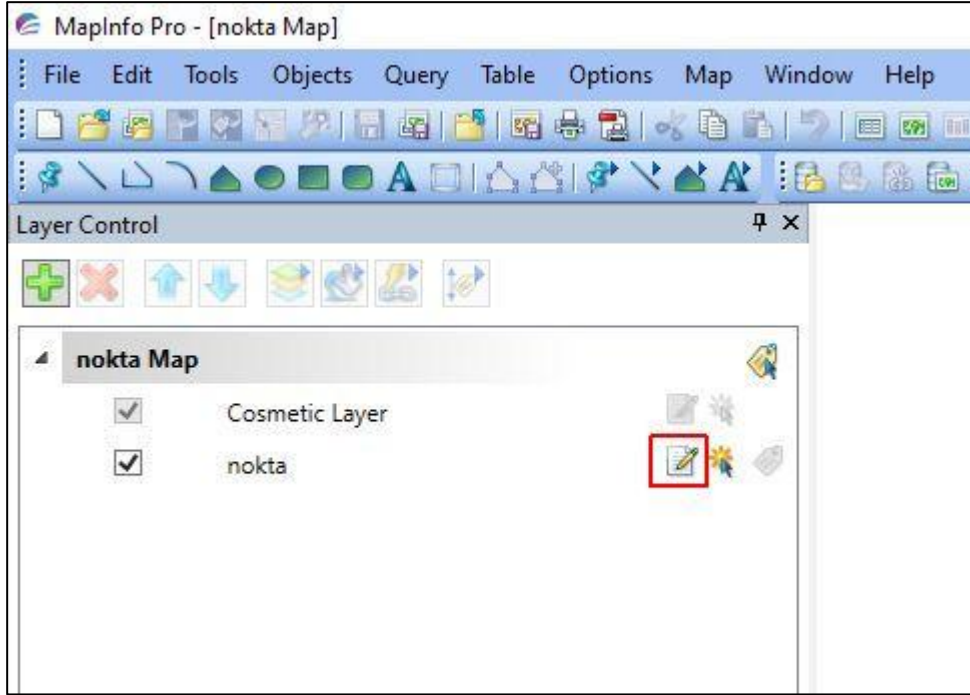
Döngü sayesinde dosyanın her bir satırına ulaşılır ve *instr()* ve *mid\$()* fonksiyonları sayesinde satır içinde yer alan noktaya ait verilere ulaşılır. Elde edilen veriler değişkenlere aktarılır. Şekil 140 29. satırda nokta bilgilerine dair verilere sahip değişkenler kullanılarak, MapInfo yazılımında oluşturulan *nokta* adlı tabakaya nokta grafik objesi olarak aktarılması sağlanır.

```

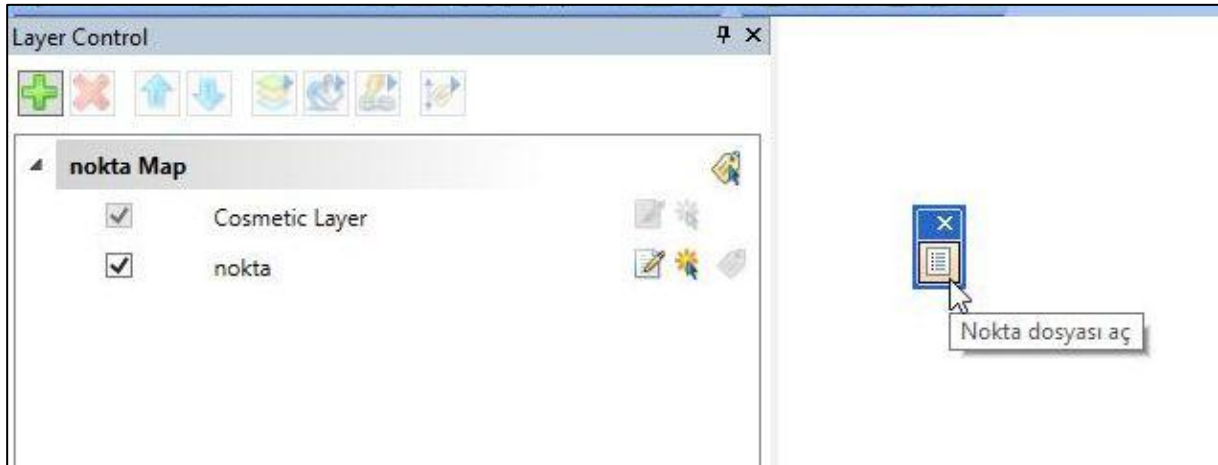
1 include "mapbasic.def"
2 include "icons.def"
3 declare sub main
4 declare sub dosya_ac
5 sub main
6     create buttonpad "Araçlar" as
7         pushbutton
8         icon MI_ICON_TABLE_LIST
9         helpmsg "Nokta dosyası aç\nNokta dosyası aç"
10        calling dosya_ac
11 end sub
12 sub dosya_ac
13     set coordsys window frontwindow()
14     dim satir,nok_ad_oku as string
15     dim bos1 as integer
16     dim dosya_ad as string
17     dim x_oku,y_oku as float
18     dosya_ad=fileopendlg("", "", "TXT", "Open Table")
19     open file dosya_ad for input as #1
20     do while not EOF(1)
21         bos1=0
22         line input #1, satir
23         if not EOF(1) then
24             bos1=instr(1,satir," ")
25             nok_ad_oku= mid$(satir,1, (bos1-1))
26             y_oku=val(mid$(satir, (bos1+1),10))
27             x_oku=val(mid$(satir, (bos1+12),11))
28             print bos1+" "+ nok_ad_oku +" "+y_oku+" "+x_oku
29             insert into nokta(nokta_ad,obj) values(nok_ad_oku,createpoint(y_oku,x_oku))
30         end if
31     loop
32     close file #1
33 end sub

```

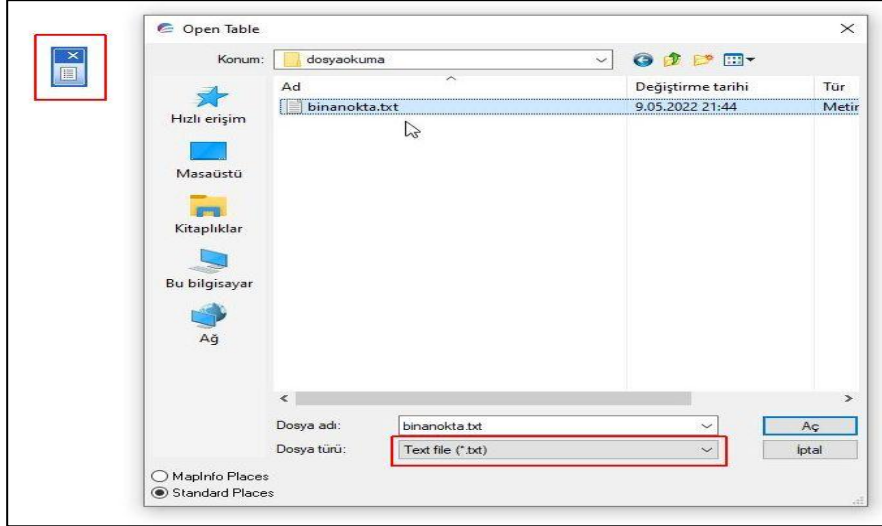
Şekil 140



Şekil 141

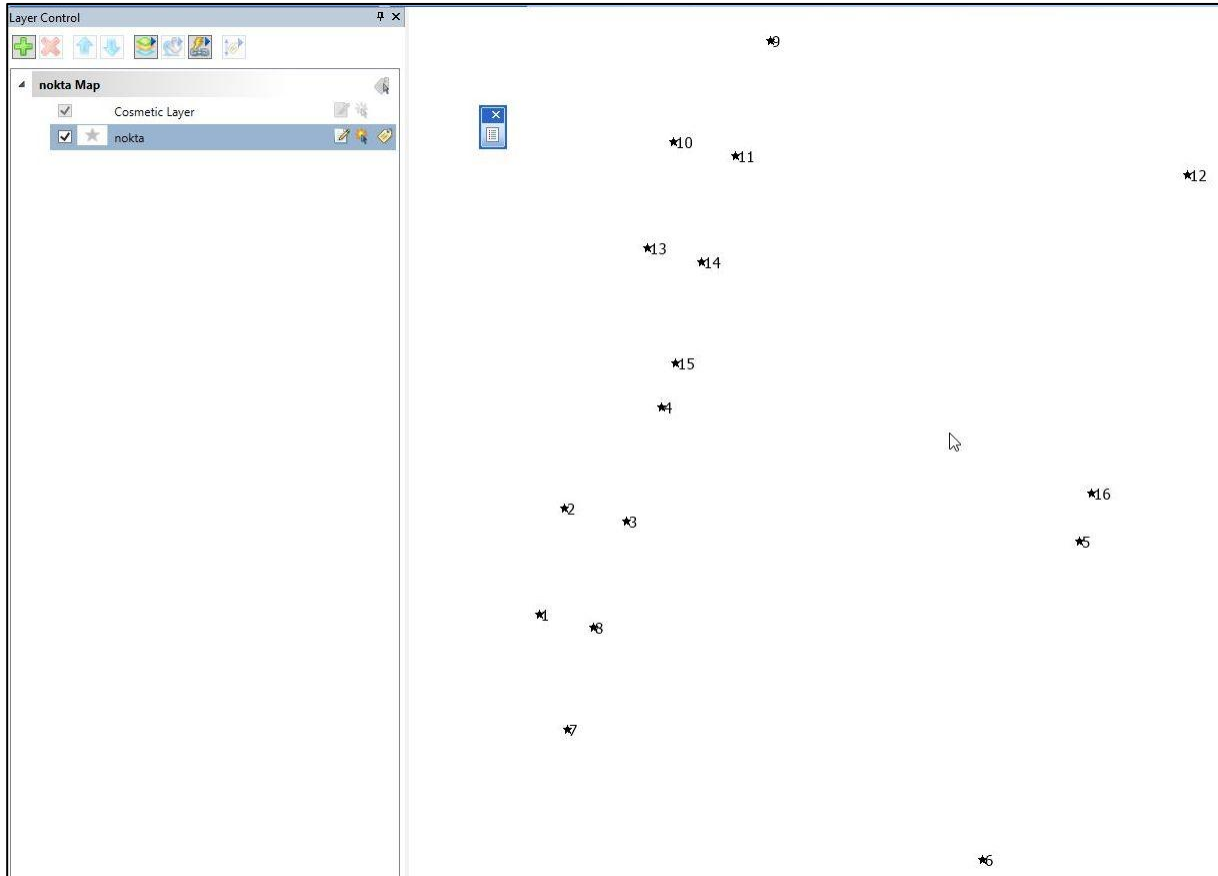


Şekil 142



Şekil 143

Şekil 144 noktaların haritaya aktarılması sonucu oluşan şekli göstermektedir. Noktalar yıldız sembolü ile temsil edilmektedir.



Şekil 144

MapInfo Yazılımında Oluşturulmuş Haritanın Tablolarından Veri Çekilmesi

Coğrafi bilgi sistemlerinin temel mantığı, coğrafik objeler harita düzleminde grafik objeler ile temsil edilir ve coğrafik objelerin öznitelikleri (objeyi niteleyen bilgiler) de grafik obje ile ilişkili tablolarda kayıt altına alınır. Gerektiğinde grafik objeler üzerinden analizler veya sorgulamalar yapılabilir, gerektiğinde de grafik objenin öznitelik bilgileri üzerinden analizler veya sorgulamalar yapılabilir.

Grafik objeleri niteleyen ve grafik haritalar ile ilişkili olan tablolar veri tabanı yapılarının temel yapı taşlarıdır. Tablo tekil kullanılabildiği gibi farklı tablolar arasında ilişkiler kurulabilir. Tablolar sadece MapInfo yazılımının da kullanılmaz. Temelinde bilgi olan tüm bilgi sistemlerinde, veri tabanı uygulamalarında kullanılır. Tabloda tutulan öznitelik verileri üze yapılacak sorgular için SQL (Structure Query Language – Yapısal Sorgulama Dili) kullanılır. SQL cümlecikleri yardımıyla

- Tablolardaki kayıtlar seçilebilir,
- Tablodaki kayıtlara yeni kayıt eklenebilir,
- Tablodaki kayıtlarda yapılacak değişiklikler güncellenebilir,
- Tablolardaki kayıtlar silinebilir.

SQL Parametreleri

SQL cümleciklerini oluşturmak için sabit parametreler vardır. Cümle yapısını oluşturmak için, cümlenin ilk parametresi gereklidir. İlk parametre yapılacak işleme (seçim, ekleme, güncelleme, silme) göre belirlenir. Her işlem kendi başlığında konu bazlı anlatılacaktır.

Tablodan Veri Seçimi İçin SQL cümlesi Kurulması (Select Parametresi)

Tablodan seçim işleminin yapılması için *Select* parametresi kullanılır.

Select *Tablodan_Seçilecek_Parametre*

From *Seçim_Yapılacak_Tablo*

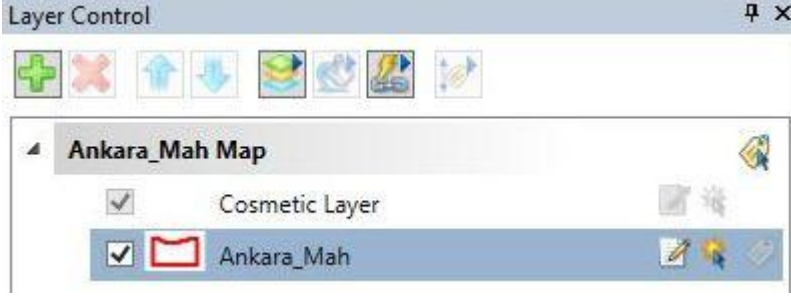
Where *Seçimde_Uygulanacak_Kısıtlamalar*

Yukarıda seçim işleminin yapılması için kullanılacak temel sorgulama yapısı bulunmaktadır.

- *Select* parametresinden sonra tablo içinde seçim yapılacak tablo öznitelik sahalarının isimleri (veya tek bir saha ismi) aralarına virgül konarak yazılır.
- *From* parametresinden sonra sorgulamada kullanılacak olan tablo veya tabloların adları yazılır.
- *Where* parametresinden sonra sorgulamada yapılacak kısıtlamalar yazılır.

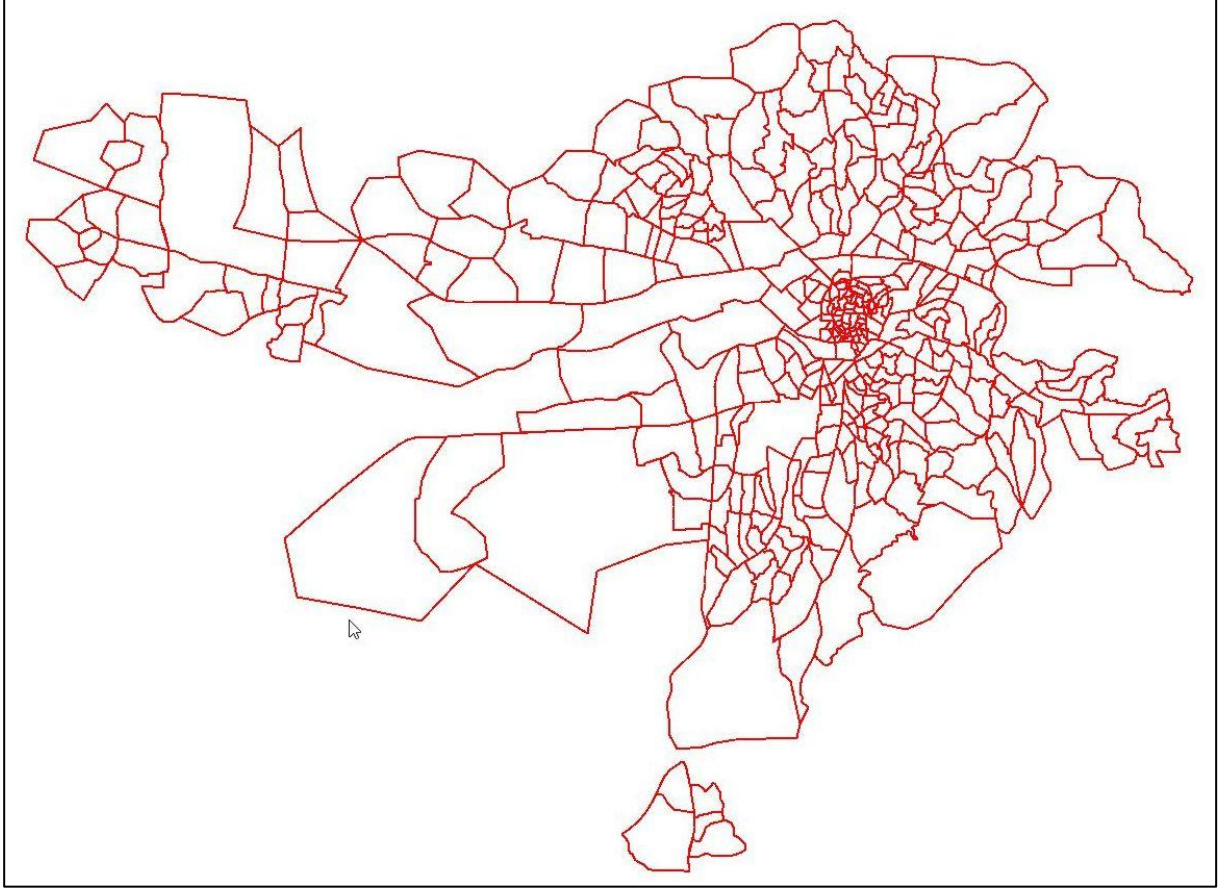
Seçim işleminin daha anlaşılır hale gelmesi için örnekler yapılacaktır. Anlatımda kullanılan örnekler Başarsoft Yazılım firmasının eğitim amaçlı hazırladığı verilerden oluşmaktadır.

Örnekte Ankara ili sınırları içindeki ilçelere bağlı mahalle coğrafik objelerinin olduğu Ankara_Mah tabakası kullanılmıştır. Şekil 145 tabakanın MapInfo yazılımında açıldığının temsildir.



Şekil 145

Mahalleler alan grafik objeleriyle temsil edilmektedir. Şekil 146 mahalle tabakasının limit bul işlemi sonrası gösteriminin temsildir.



Şekil 146

Şekil 147 harita penceresinde yaklaşma işlemi sonucunda oluşan görüntüdür. Haritada etiketler de açılmış ve mahalle isimleri gözükmemektedir.



Şekil 147

Şekil 148 grafik objelerin öznelik bilgilerinin tutulduğu tablonun temsilidir. Tablo incelendiğinde Mahalle objelerine ait sahalara görülmektedir. Tablo mahalle adı bilgisi (MAHALLE_ADI sahasında tutulan bilgi), mahallenin içinde olduğu ilçe adı bilgisi (ILCE_ADI sahasında tutulan bilgi), ilçelerin sorgulamalarını kolaylaştırmak ve Ankara içindeki ilçelerin tanımlamalarını tekil hale getirmek için kullanılan ilçe kod bilgisi (ILCE_KOD sahasında tutulan bilgi), Ankara içindeki ilçelerin sınırları içindeki mahalleleri tekil hale getirmek ve sorgulamalarını kolaylaştırmak için oluşturulan mahalle kod bilgisi (MAH_KODU sahasında tutulan bilgi) sahalardan oluşur.

MAHALLE_ADİ	İLCE_ADİ	İLCE_KODU	MAH_KODU
Kent Koop Mah	YENİMAHALLE	5	52
Meşrutiyet Mah	CANKAYA	1	1
Yüce-tepe Mah	CANKAYA	1	15
Mebusevleri- Anıtkabir	CANKAYA	1	14
Namık Kemal Mah	CANKAYA	1	16
Kızılay Mah	CANKAYA	1	9
Y.bahçelievler Mah	CANKAYA	1	19
Kocatepe Mah	CANKAYA	1	21
Cumhuriyet Mah	CANKAYA	1	20
Fidanlık Mah	CANKAYA	1	24
Sağlık Mah	CANKAYA	1	25
Kavaklıdere Mah	CANKAYA	1	2
Esatoğlu Mah	CANKAYA	1	39
Tınaztepe Mah	CANKAYA	1	38

Şekil 148

Şekil 149 tablodan seçilen bir kayıt ile grafik objenin de seçili olduğunun temsilidir.

MAHALLE_ADİ	İLCE_ADİ	İLCE_KODU	MAH_KODU
S.cengiz Karaca Mah	CANKAYA	1	83
Malazgirt Mah	CANKAYA	1	64
Ata Mah	CANKAYA	1	58
Huzur Mah	CANKAYA	1	59
Cevizlidere Mah	CANKAYA	1	52
100.yıl İşçi Blokları	CANKAYA	1	89
Yukarı Öveçler Mah	CANKAYA	1	99
Asağı Öveçler Mah	CANKAYA	1	98
Sokullu Mehmet Paşa Mah	CANKAYA	1	97
İçkale Mah	ALTINDAG	3	28
İzzettin Mah	ALTINDAG	3	14
Yeğenbey Mah	ALTINDAG	3	26
Yenice Mah	ALTINDAG	3	27
Keklikpınarı Mah	CANKAYA	1	63
Karapınar Mah	CANKAYA	1	61
Gökkuşuğı Mah	CANKAYA	1	60

Şekil 149

Örnek 1:

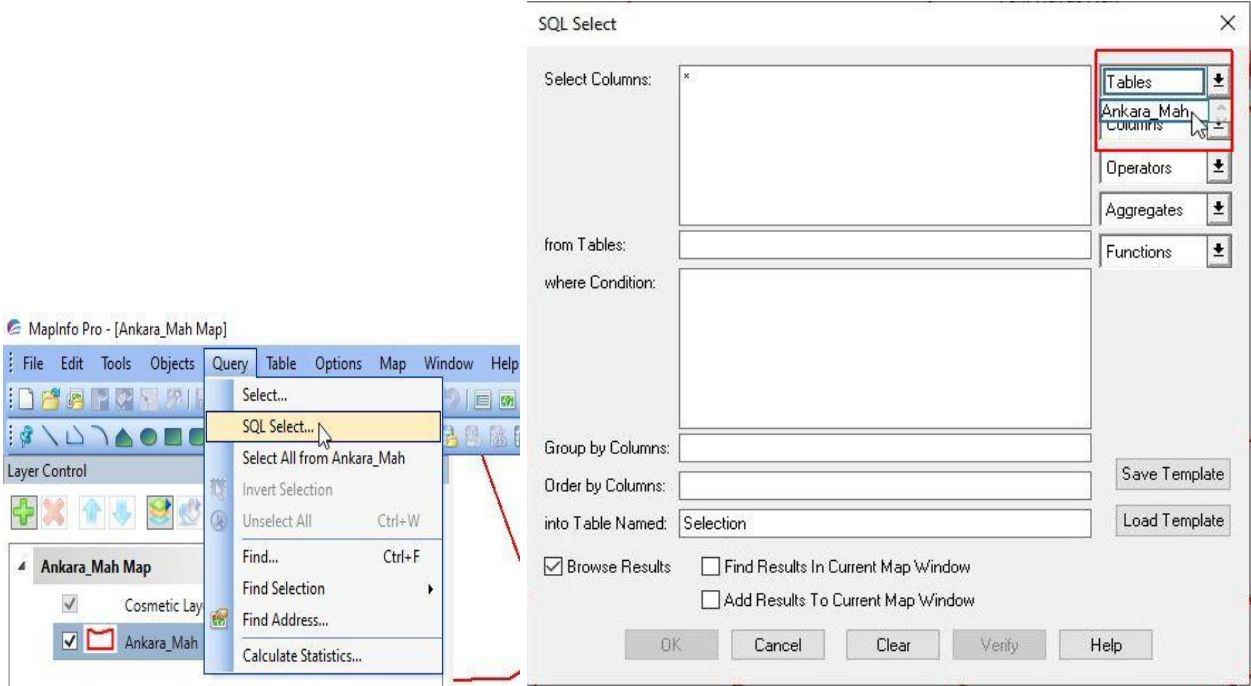
Ankara_Mah isimindeki tablodaki tüm kayıtların seçimini yapan sorgu cümlesini yazınız

Select *

From Ankara_mah

Select seçim parametresinde kullanılan yıldız tüm sahalardaki verilerin seçilmesini ifade eder. From parametresinde tabakanın adı kullanılmıştır. Where parametresi kullanılmadığı için tüm kayıtlar seçilecektir. Aynı cümlecik aşağıdaki gibi de yazılabilir.

Select * From Ankara_mah



Şekil 150

SQL Select

Select Columns: *

from Tables: Ankara_Mah

where Condition:

Group by Columns:

Order by Columns:

into Table Named: Selection

Browse Results Find Results In Current Map Window
 Add Results To Current Map Window

OK Cancel Clear Verify Help

Şekil 151

Query1 Browser

MAHALLE ADI	ILCE ADI	ILCE_KODU	MAH_KODU
Kent Koop Mah	YENIMAHALLE	5	52
Megrutiyet Mah	CANKAYA	1	1
Yuce-tepe Mah	CANKAYA	1	15
Mebusevlen- Anitkabir	CANKAYA	1	14
Namik Kemal Mah	CANKAYA	1	16
Kizilay Mah	CANKAYA	1	9
Y.bahcelievler Mah	CANKAYA	1	19
Kocatepe Mah	CANKAYA	1	21
Cumhuriyet Mah	CANKAYA	1	20
Fidanlik Mah	CANKAYA	1	24
Saglik Mah	CANKAYA	1	25
Kavaklidere Mah	CANKAYA	1	2
Esatoglu Mah	CANKAYA	1	39
Tinaztepe Mah	CANKAYA	1	38
Seyran Mah	CANKAYA	1	35
Dogus Mah	CANKAYA	1	37
Zafertepe Mah	CANKAYA	1	36
Gokturk Mah	CANKAYA	1	105
Ozgurluk - Metin Oktay Mah	CANKAYA	1	103
Umut Mah	CANKAYA	1	75
Acikpasa Mah	CANKAYA	1	74
Mimarsinan Mah	CANKAYA	1	106
Arkatopraklik Mah	CANKAYA	1	32
Dilekler Mah	CANKAYA	1	31
Sehit Cengiz Topel Mah	MAMAK	2	11
Ellinci Yil Mah	CANKAYA	1	78
Çamlitepe Mah	CANKAYA	1	26
Fakulteler Mah	CANKAYA	1	27
Ertugrugazi Mah	CANKAYA	1	29
Cebeci Mah	CANKAYA	1	28
Abidinpasa Mah	MAMAK	2	3
Aşik Veysel Mah	MAMAK	2	8

Ankara_Mah Map

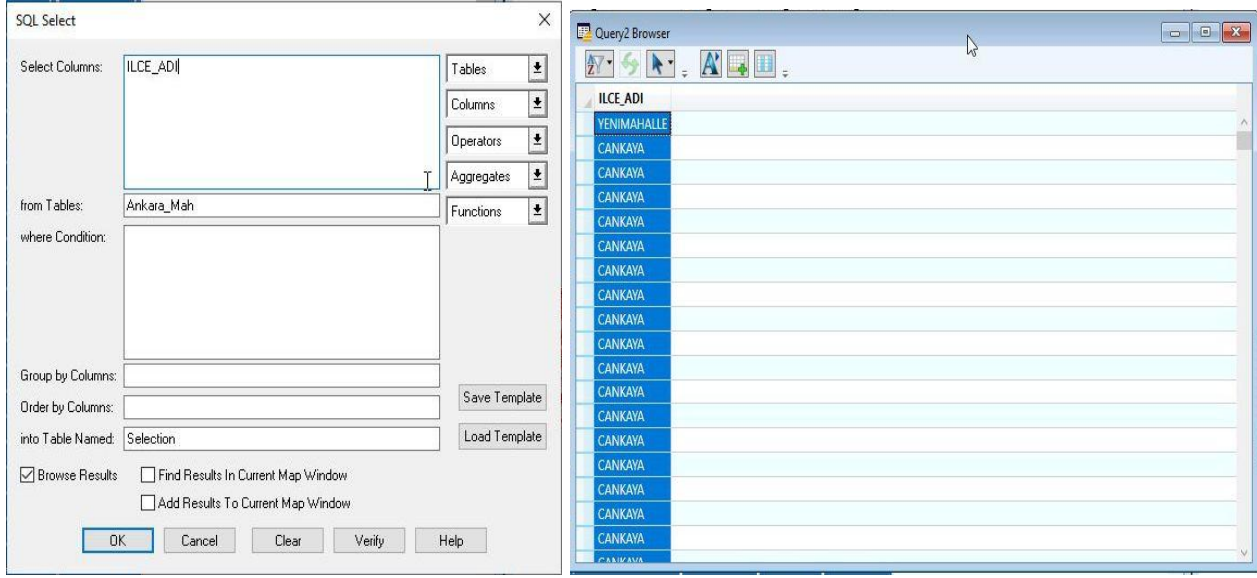
Şekil 152

Örnek 2:

Ankara_Mah isimindeki tabloda ILCE_ADİ sahasındaki kayıtlı ilçelerin seçilmesi için SQL cümlesi yazınız.

Select ILCE_ADİ

From Ankara_mah



Şekil 153

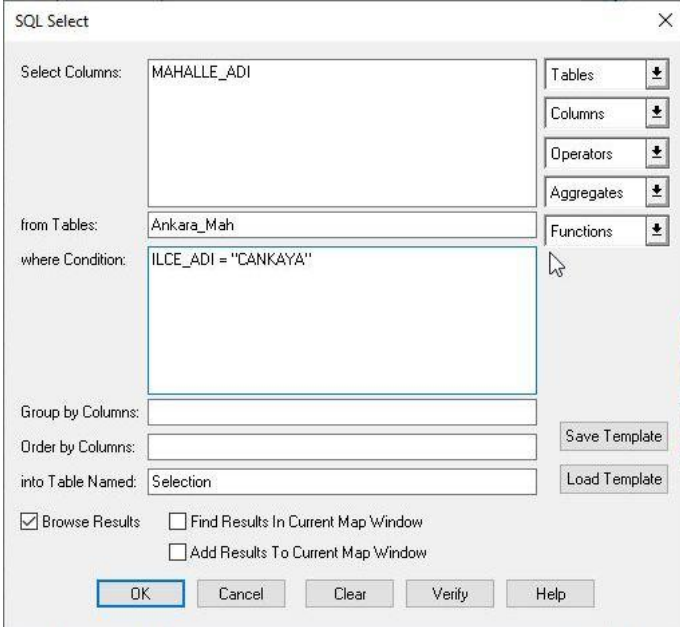
Örnek 3:

Ankara_mah tablosunda Çankaya ilçesindeki mahallelerin listesini sorgulayan SQL cümlecğini yazınız

Select MAHALLE_ADI

From Ankara_mah

Where ILCE_ADI= "CANKAYA"



Şekil 154

MAHALLE_ADI	ILCE_ADI	ILCE_KODU	MAH_KODU
Kent Koop Mah	YENIMAHALLE	5	52
Meşrubiyet Mah	CANKAYA	1	1
Yüce-tepe Mah	CANKAYA	1	15
Mebusevleri- Anıtkabir	CANKAYA	1	14
Namık Kemal Mah	CANKAYA	1	16
Kızılay Mah	CANKAYA	1	9
Y.bahçelievler Mah	CANKAYA	1	19
Kocatepe Mah	CANKAYA	1	21
Cumhuriyet Mah	CANKAYA	1	20
Fidanlık Mah	CANKAYA	1	24
Sağlık Mah	CANKAYA	1	25
Kavaklıdere Mah	CANKAYA	1	2
Esatoğlu Mah	CANKAYA	1	39
Tinaztepe Mah	CANKAYA	1	38
Seyran Mah	CANKAYA	1	35
Doğuş Mah	CANKAYA	1	37
Zafer-tepe Mah	CANKAYA	1	36
Göktürk Mah	CANKAYA	1	105
Özgürlük - Metin Oktay Mah	CANKAYA	1	103
Umut Mah	CANKAYA	1	75
Aşıkpaşa Mah	CANKAYA	1	74
Mimarsinan Mah	CANKAYA	1	106
Arkatopraklık Mah	CANKAYA	1	32
Dilekler Mah	CANKAYA	1	31
Şehit Cengiz Topel Mah	MAMAK	2	11
İlinci Yıl Mah	CANKAYA	1	78
Çamli-tepe Mah	CANKAYA	1	26
Fakülte-ler Mah	CANKAYA	1	27
Ertuğrul-gazi Mah	CANKAYA	1	29
Cebeci Mah	CANKAYA	1	28
Abidinpaşa Mah	MAMAK	2	3
Akık Vey-sel Mah	MAMAK	2	8
Kartal-tepe Mah	MAMAK	2	5
Mehtap Mah	MAMAK	2	7
Çağ-layan Mah	MAMAK	2	6
Tuzlu-çayır Mah	MAMAK	2	17

MAHALLE_ADI
Yüce-tepe Mah
Mebusevleri- Anıtkabir
Namık Kemal Mah
Kızılay Mah
Y.bahçelievler Mah
Kocatepe Mah
Cumhuriyet Mah
Fidanlık Mah
Sağlık Mah
Kavaklıdere Mah
Esatoğlu Mah
Tinaztepe Mah
Seyran Mah
Doğuş Mah
Zafer-tepe Mah
Göktürk Mah
Özgürlük - Metin Oktay Mah
Umut Mah
Aşıkpaşa Mah
Mimarsinan Mah
Arkatopraklık Mah
Dilekler Mah
İlinci Yıl Mah
Çamli-tepe Mah
Fakülte-ler Mah
Ertuğrul-gazi Mah
Cebeci Mah
Barbaros Mah

Şekil 155

Örnek 4:

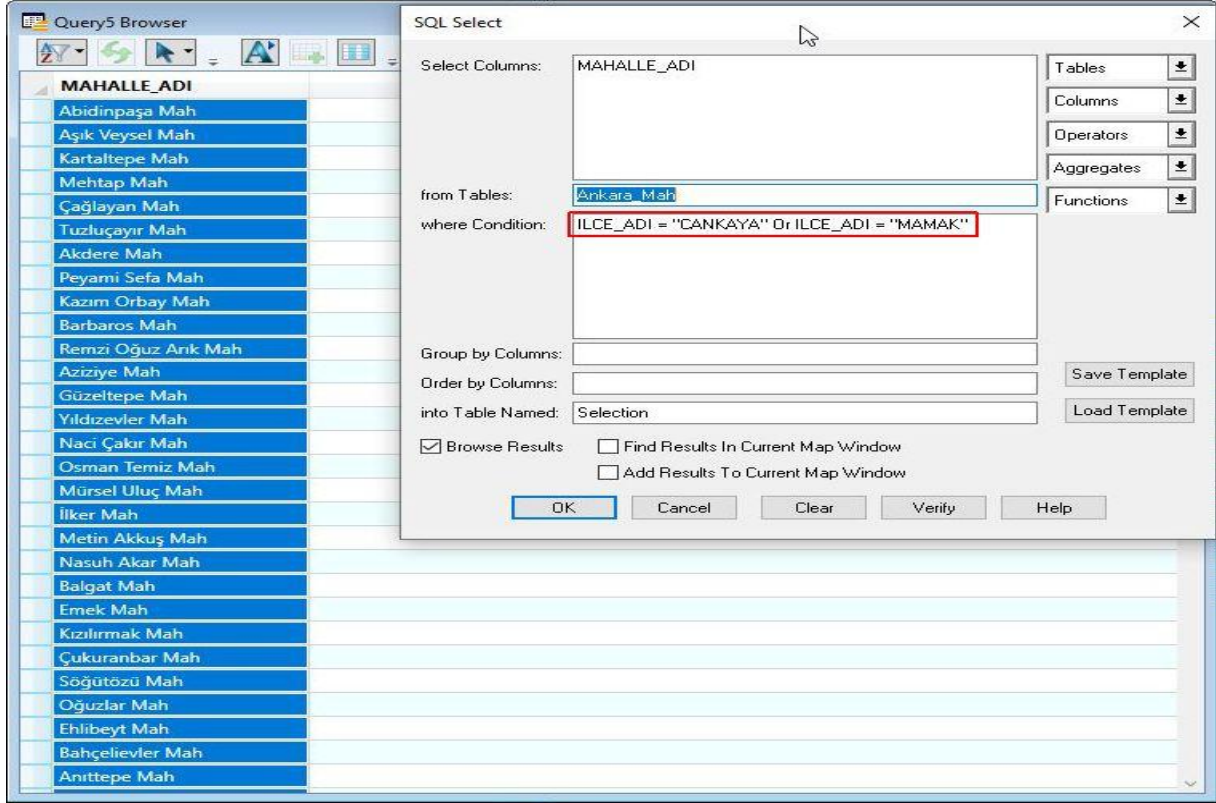
Ankara_mah tabakasında Çankaya ile Mamak ilçelerindeki mahallelerin isimlerini listeleyen SQL cümleciğiniz yazınız.

Sorgulamada hem Çankaya ilçesinde olan mahallelerin adları isteniyor, hem de Mamak ilçesindeki mahallelerin adları isteniyor. Veri kaydı yapılırken ilçe adı ve ilçedeki mahallenin adı kaydı yapılmakta. Sorgulamanın yapılması için OR (veya) operatörü kullanılmak zorunda. AND (ve) operatörü kullanamaz. Çünkü hem kayıt işleminde tek bir ilçe ve o ilçedeki tek bir mahalle kaydı yapılmaktadır.

```
Select MAHALLE_ADI
```

```
From Ankara_mah
```

```
Where ILCE_ADI= "CANKAYA" or ILCE_ADI= "MAMAK"
```



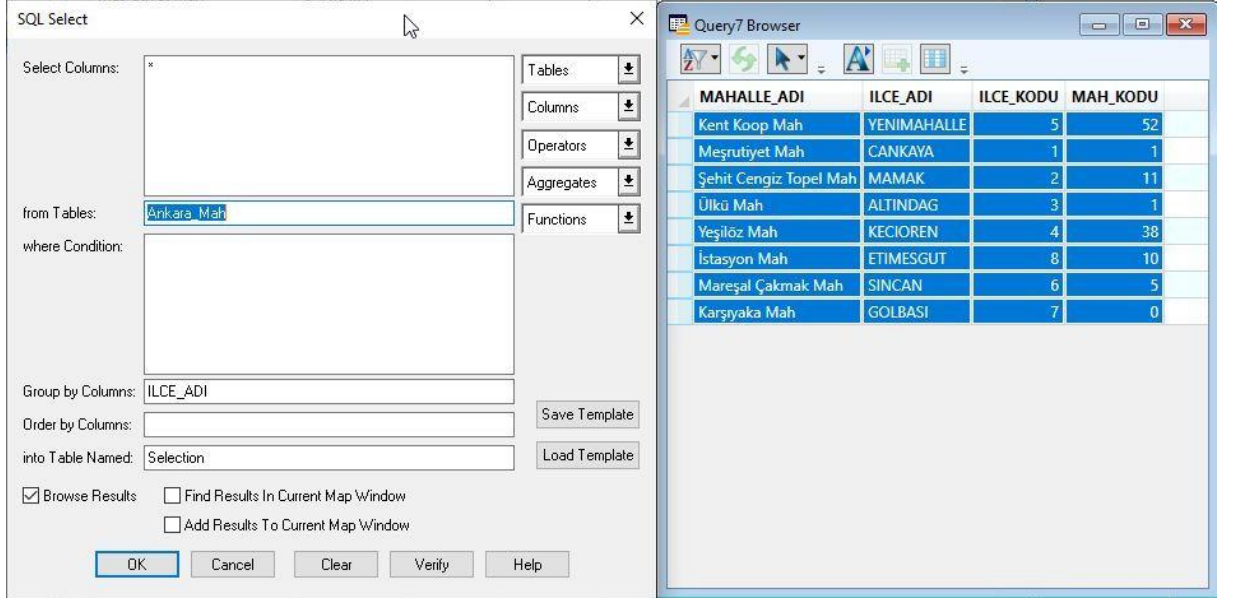
Tablo Verilerinin Sahalara Göre Gruplandırılması

SQL sorgu cümlecği ile yapılan sorgulama sonuçlarının gruplandırılması yapılabilir. Gruplandırma, bir sahadaki verilerin arasında tekil olanların grup haline getirilmesini sağlar. Gruplandırma işlemi için *Group by* sql komutu kullanılır.

Örnek 6:

Ankara_mah tablosu kayıtları ilçe adlarına göre gruplandırılması için gerekli SQL cümlecğini yazınız.

```
Select *
From Ankara_mah
Group by ILCE_AD
```



Şekil 156

Sorgu cümleciğinin sonucu Şekil 156 sağ resimde gözükmektedir. Select parametresinde * karakteri girildiği için diğer sahalarda veriler gözükmemektedir. Sonuç incelediğinde ilçe adı sütunundaki veriler için kayıtlar tekil olacak şekilde incelenmiş ve tekil olan değerler gruplandırılmıştır.

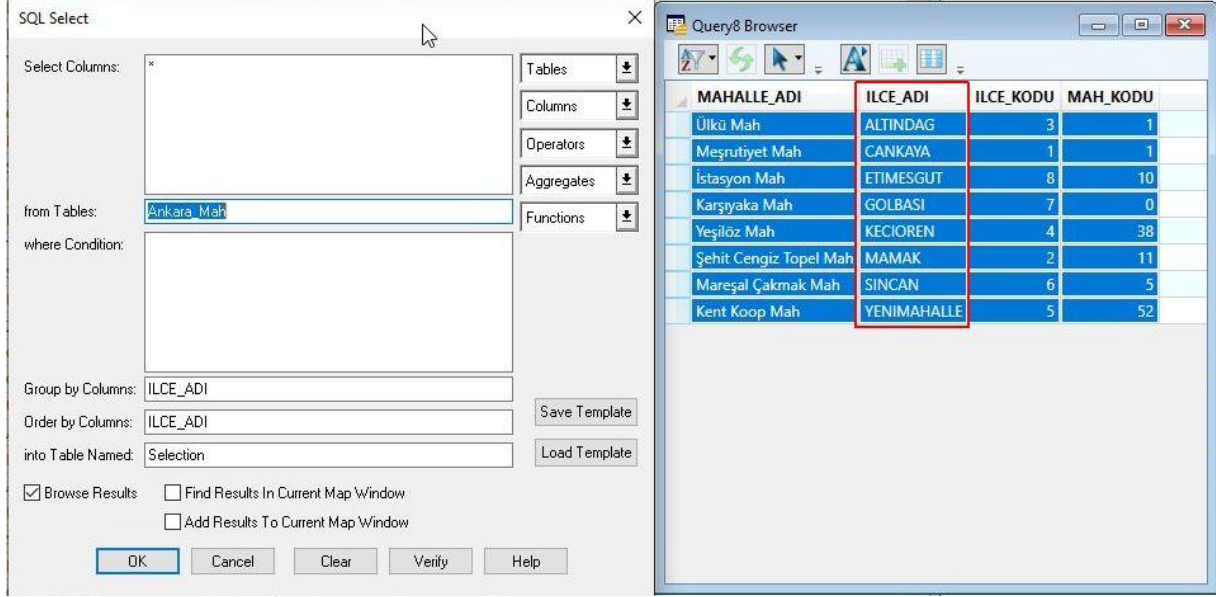
Tablodaki Verilerin Sahalara Göre Sıralanması

Tabloda var olan veriler seçimi yapılırken bir sahaya göre veriler büyükten küçüğe veya küçükten büyüğe sıralanabilir. Sıralama işlemi özellikle kullanıcı için yapılacak diyaloglara veri aktarımında önem kazanacaktır. Sıralama işlemi için Order By sql komutu kullanılır.

Örnek 7:

Ankara_mah tablosunda ilçe adları hem gruplandırılсын, hem de yapılan gruplandırma ilçe adlarına göre sıralansın.

```
Select *
From Ankara_mah
Group By ILCE_AD
Order By ILCE_AD
```

Şekil 157

Şekil 157 sağ resim sonuç seçimleri göstermektedir. Sonuç tablo incelendiğinde kayıtlar hem ilçe adlarına göre gruplandırılmış hem de ilçe adlarına göre sıralanmıştır.



SQL komutlarından seçim işlemleri dışındaki silme, güncelleme işlemleri sonraki konularda örnekler üzerinde işlenecektir. Daha önceki konularda ekleme işlemleri konu içinde işlenmiştir.

SQL Parametrelerinin Mapbasic Kodları İçinde Kullanımı

Mapbasic kodları ile yapılan örneklerde MapInfo yazılımı harita ekranındaki objeler için grafik objelerin eklenmesi, objeler hakkında bilgi alınması ve objeler koordinatları üzerinde düzenleme işlemleri yapılmıştır. Sorgulama işlemleri sayesinde grafik objelerin öznelik bilgilerinin olduğu tablolarda sorgu yapılabilecek. Sorgulama sonuçları gerektiğinde diyalog araçları içine doldurulup son kullanıcıya yönelik programlar yazılabilecek.

Örnek 8:

Ankara_mah tablosunda ilçe ad bilgileri bir listbox diyalog aracına doldurulacak. Aynı listbox aracından seçilecek ilçe adı değeri alınarak, o ilçe içerisindeki mahalle adları ikinci bir listbox aracına doldurulması istenmektedir

```
1 include "mapbasic.def"
2 declare sub main I
3 declare sub mahalle_doldur
4 dim ilce_ad(1), mah_ad(1) as string
5 sub main
6     dim k as integer
7     select * from Ankara_mah group by ILCE_ADI order by ILCE_ADI into sel
8     redim ilce_ad(tableinfo(sel,TAB_INFO_NROWS))
9     fetch first from sel
10    k=1
11    do while not EOT(sel)
12        ilce_ad(k)=sel.ILCE_ADI
13        k=k+1
14        fetch next from sel
15    loop
16
17    select * from Ankara_mah where ILCE_ADI=ilce_ad(1) order by MAHALLE_ADI into sel
18    redim mah_ad(tableinfo(sel,TAB_INFO_NROWS))
19    k=1
20    fetch first from sel
21    do while not EOT(sel)
22        mah_ad(k)=sel.MAHALLE_ADI
23        k=k+1
24        fetch next from sel
25    loop
26
27 Dialog
28     Title "İlçe-Mahalle Bilgileri"
29     Width 217 Height 97
30
31 Control ListBox
32     Id 1
33     Position 8, 16
34     Width 82 Height 59
```

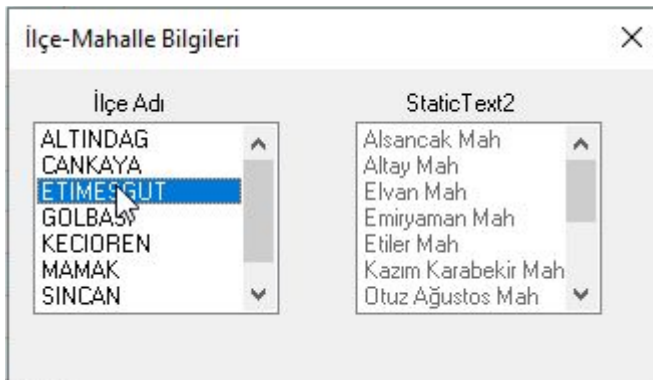
Şekil 158

```

35 title from variable ilce_ad
36 calling mahalle_doldur
37 Control ListBox
38 Id 2
39 Position 115, 16
40 Width 82 Height 59
41 Disable
42 title from variable mah_ad
43 Control StaticText
44 Position 28, 6
45 Width 29 Height 8
46 Title "İlçe Adı"
47 | I
48 Control StaticText
49 Position 132, 6
50 Width 41 Height 8
51 Title "Mahalle Adı"
52 end sub
53 sub mahalle_doldur
54 dim i,k as integer
55 i=readcontrolvalue(1)
56 select * from Ankara_mah where ILCE_ADI=ilce_ad(i) order by MAHALLE_ADI into sel
57 redim mah_ad(tableinfo(sel,tab_info_nrows))
58 k=1
59 fetch first from sel
60 do while not EOT(sel)
61 mah_ad(k)=sel.MAHALLE_ADI
62 k=k+1
63 fetch next from sel
64 loop
65 alter control 2 title from variable mah_ad
66 end sub

```

Şekil 159



Şekil 160

Birden Fazla Seçilen Alan Objelerinin Alan Bilgilerini Listelemek
Seçilen Alan Objesi Etrafına Nokta Adları Oluşturmak
Yan Nokta Hesabı Yapan Buton

Menu Tasarımı

MapBasic programlama dili kullanılarak MapInfo yazılımı menülerine ek kendi amaçlarımız için menü ve alt menüler ekleyebilir, bu menüler sayesinde kendi ihtiyaçlarımızı karşılayacak işlemler tasarlanabilir.

Şekil 41 menü tasarımına ait örnek verilmiştir. Menu oluşturulmasına dair kod 6 – 8. Satırlar arasında bulunmaktadır.

Create Menu “Menu adı” *As* → verilen menü adıyla menü oluştur

“Alt Menu” *Calling* *Yordam* → verilen alt menü adıyla alt menü oluştur ve tıklandığında ilgili yordamı çağır.

Alter Menu Bar Add “Menu Adı” → var olan Menü çubuğuna oluşturulan menüyü ekle

```

menutasarımı.mb
1  Include "MapBasic.def"
2
3  Declare Sub Main
4  Declare sub ekran_yaz
5  Sub Main()
6      create menu "Harita Araçları" as
7          "Ekrana Yaz" Calling ekran_yaz
8      Alter menu bar add "Harita Araçları"
9  End Sub
10
11 sub ekran_yaz
12     note "Menu yazısı"
13 End Sub

```

Şekil 161

Örnekten *Harita Araçları* adı altında menü ve *Ekran Yaz* adlı alt menü oluşturulmasına dair menü tasarımı gözükmektedir. *Ekran Yaz* menüsü seçildiğinde *ekran_yaz* adlı yordam çalışması için kod tasarlanmıştır. Şekil 42 örnek kodun derlenip çalıştırılmasıyla oluşan görüntüyü temsil etmektedir. Şekil 42 sol resimde menüyü göstermekte, şekil 42 sağ resim ise *Ekran Yaz* menüsü seçildiğinde ekrana çıkması tasarlanmış mesaj kutusu gözükmektedir.



Şekil 162

Döngü İşlemleri

Ekler

Haritadaki Grafik Objeler ile Çalışılırken Bilinmesi Gereken Fonksiyonlar

Mapinfo programında eklenen tabakalar üzerinde çalışırken veya tabaka içinde var olan grafik objelerle çalışırken bazı hazır Mapbasic fonksiyonlarının bilinmesi gerekir.

Set Coordsys= komut satırı kullanılan harita penceresinin koordinat, birim, projeksiyon bilgilerinin Mapbasic kodu içinde de geçerli olmasını sağlar.

Set Coordsys window frontwindow() → Çizim sıralamasında en önde yer alan pencerenin koordinat sisteminin Mapbasic kodu içinde geçerli olmasını sağlar.

MapperInfo()= Harita penceresi hakkında bilgi geri döndüren fonksiyondur.

Kullanımı: MapperInfo(Pencere_id, bilgi)

Bilgi İçin Kullanılacak kod	Geri Dönen Özellik
MAPPER_INFO_AREAUNITS	Haritanın alan birimi geri döner
MAPPER_INFO_CENTERX	Harita Merkezinin X değeri geri döner.
MAPPER_INFO_CENTERY	Harita Merkezinin Y değeri geri döner
MAPPER_INFO_COORDSYS_CLAUSE	Koordinat sisteminin ismi geri döner
MAPPER_INFO_COORDSYS_NAME	Haritanın projeksiyon bilgisi, MAPINFOW.PRJ dosyasındaki ismiyle geri döner.
MAPPER_INFO_DISPLAY	Tam sayı veri döndürür. Durum çubuğundaki harita ölçeği, koordinat bilgisi, yakınlık (zoom) bilgilerinden hangisinin gösterildiğini belirten sayısal değer döner.
MAPPER_INFO_DISPLAY_DMS	Haritada koordinatların gösterim şekline ait sayısal değer döner. • MAPPER_INFO_DISPLAY_DECIMAL for degrees decimal coordinates (0) • MAPPER_INFO_DISPLAY_DEGMINSEC for degrees, minutes, seconds coordinates (1) • MAPPER_INFO_DISPLAY_MGRS for Military Grid System coordinates (2)
MAPPER_INFO_DISTUNITS	Harita biriminin gösterim şekli geri döner.
MAPPER_INFO_EDIT_LAYER	Haritanın düzenlenebilir olduğuna dair bilgi geri döndürür. A value of zero means that the Cosmetic layer is editable. A value of -1 means that no layer is editable.
MAPPER_INFO_LAYERS	Harita penceresindeki tabakanın numarasını geri döndürür.
MAPPER_INFO_MAXX	Harita penceresindeki en büyük X koordinat bilgisini geri döndürür.
MAPPER_INFO_MAXY	Harita penceresindeki en büyük Y koordinat bilgisini geri döndürür.
MAPPER_INFO_MINX	Harita penceresindeki en küçük X koordinat bilgisini geri döndürür.
MAPPER_INFO_MINY	Harita penceresindeki en küçük Y koordinat bilgisini geri döndürür.

MAPPER_INFO_NUM_THEMATI C	Haritadaki tematik tabaka sayısını geri döndürür
MAPPER_INFO_SCALE	
MAPPER_INFO_SCROLLBARS	
MAPPER_INFO_XYUNITS	
MAPPER_INFO_ZOOM	
MAPPER_INFO_COORDSYS_CL AUSE _WITH_BOUNDS	
MAPPER_INFO_MOVE_DUPLIC ATE_ NODES	
MAPPER_INFO_DIST_CALC_TY PE	
MAPPER_INFO_CLIP_REGION	
MAPPER_INFO_CLIP_TYPE	

Frontwindow()= Görünüm sıralamasında en önde olan haritanın numarasını tamsayı olarak döndürür.

```
Dim pencere_id, harita_b as integer
pencere_id = frontwindow()
harita_b = mapperinfo(pencere_id, MAPPER_INFO_EDIT_LAYER)
```

Yukarıdaki Mapbasic program kod satırlarında, *pencere_id* adlı değişkene görünüm sıralamasında en öndeki pencerenin id değeri aktarıldı. *MapperInfo()* fonksiyonunun ilk parametresinde *pencere_id* parametresi kullanıldı. Kod satırında *MapperInfo()* fonksiyonunun ikinci parametresi *MAPPER_INFO_EDIT_LAYER*, Mapbasic.def dosyasında tanımlanmış olan bir parametredir. Bu parametre sayesinde id değeri verilen tabakanın düzenlenebilir olup olmadığı sorgulanıyor. Bu sorgulama Mapbasic program kodu ile grafik obje ekleneceği zaman yapılmaktadır.

LayerInfo()= Haritada kullanılan tabaka hakkında bilgi geri döndüren fonksiyondur.

<i>Tabaka Hakkında İstenilen Bilgi</i>	<i>Geri Dönene Değer</i>
LAYER_INFO_NAME	String indicating the name of the table associated with this map layer. If the specified layer is the map's Cosmetic layer, the string will be a table name such as "Cosmetic1"; this table name can be used with other statements (for example, Select).
LAYER_INFO_EDITABLE	Logical value; TRUE if the layer is editable.
LAYER_INFO_LBL_PARTIALSEGS	Logical value; TRUE if the Label Partial Objects check box is selected for this layer.
LAYER_INFO_SELECTABLE	Logical value; TRUE if the layer is selectable.
LAYER_INFO_PATH	String value representing the full directory path of the table associated with the map layer.
LAYER_INFO_ZOOM_LAYERED	Logical; TRUE if zoom-layering is enabled.
LAYER_INFO_ZOOM_MIN	Float value, indicating the minimum zoom value (in MapBasic's current distance units) at which the

	layer displays. (To set MapBasic’s distance units, use Set Distance Units.)
LAYER_INFO_ZOOM_MAX	Float value, indicating the maximum zoom value at which the layer displays.
LAYER_INFO_COSMETIC	Logical; TRUE if this is the Cosmetic layer.
LAYER_INFO_DISPLAY	SmallInt, indicating how and whether this layer is displayed; return value will be one of these values: <ul style="list-style-type: none"> • LAYER_INFO_DISPLAY_OFF (the layer is not displayed); • LAYER_INFO_DISPLAY_GRAPHIC (objects in this layer appear in their “default” style—the style saved in the table); • LAYER_INFO_DISPLAY_GLOBAL (objects in this layer are displayed with a “style override” specified in Layer Control); • LAYER_INFO_DISPLAY_VALUE (objects in this layer appear as thematic shading)
LAYER_INFO_OVR_LINE	Pen style used for displaying linear objects.
LAYER_INFO_OVR_PEN	Pen style used for displaying the borders of filled objects.
LAYER_INFO_OVR_BRUSH	Brush style used for displaying filled objects.
LAYER_INFO_OVR_SYMBOL	Symbol style used for displaying point objects.
LAYER_INFO_OVR_FONT	Font style used for displaying text objects.
LAYER_INFO_LBL_CURFONT	For applications compiled with MapBasic 4.0 or later, this query always returns false. For applications compiled with MapBasic 3.x, this query returns the following values: Logical value: TRUE if layer is set to use the current font, or FALSE if layer is set to use the custom font (see LAYER_INFO_LBL_FONT).
LAYER_INFO_LBL_FONT	Font style used in labels.
LAYER_INFO_LBL_EXPR	String value: the expression used in labels.
LAYER_INFO_LBL_LT	Smallint value indicating what type of line, if any, connects a label to its original location after you move the label. The return value will match one of these values: <ul style="list-style-type: none"> • LAYER_INFO_LBL_LT_NONE (no line) • LAYER_INFO_LBL_LT_SIMPLE (simple line) • LAYER_INFO_LBL_LT_ARROW (line with an arrow head)
LAYER_INFO_LBL_PARALLEL	Logical value: TRUE if layer is set for parallel labels
LAYER_INFO_LBL_POS	Smallint value, indicating label position. Return value will match one of these values (T=Top, B=Bottom, C=Center, R=Right, L=Left): <ul style="list-style-type: none"> • LAYER_INFO_LBL_POS_TL • LAYER_INFO_LBL_POS_TC • LAYER_INFO_LBL_POS_TR • LAYER_INFO_LBL_POS_CL • LAYER_INFO_LBL_POS_CC • LAYER_INFO_LBL_POS_CR • LAYER_INFO_LBL_POS_BL

	<ul style="list-style-type: none"> • LAYER_INFO_LBL_POS_BC • LAYER_INFO_LBL_POS_BR
LAYER_INFO_LBL_VISIBILITY	<p>Smallint value, indicating whether labels are visible; see the Visibility clause of the Set Map statement. Return value will be one of these values:</p> <ul style="list-style-type: none"> • LAYER_INFO_LBL_VIS_ON (labels always visible) • LAYER_INFO_LBL_VIS_OFF (labels never visible) • LAYER_INFO_LBL_VIS_ZOOM (labels visible when in zoom range)
LAYER_INFO_LBL_ZOOM_MIN	Float value, indicating the minimum zoom distance for this layer's labels.
LAYER_INFO_LBL_ZOOM_MAX	Float value, indicating the maximum zoom distance for this layer's labels.
LAYER_INFO_LBL_AUTODISPLAY	Logical value: TRUE if this layer is set to display labels automatically. See the Auto clause of the Set Map statement.
LAYER_INFO_LBL_OVERLAP	Logical value; TRUE if overlapping labels are allowed.
LAYER_INFO_LBL_DUPLICATES	Logical value; TRUE if duplicate labels are allowed.
LAYER_INFO_LBL_OFFSET	Smallint value from 0 to 50, indicating how far the labels are offset from object centroids. The offset value represents a distance, in points.
LAYER_INFO_LBL_MAX	Integer value, indicating the maximum number of labels allowed for this layer. If no maximum has been set, return value is 2,147,483,647.
LAYER_INFO_LBL_PARTIALSEGMENTS	Logical value; TRUE if the Label Partial Segments check box is checked for this layer.
LAYER_INFO_ARROWS	Logical value; TRUE if layer displays direction arrows on linear objects.
LAYER_INFO_NODES	Logical value; TRUE if layer displays object nodes.
LAYER_INFO_CENTROIDS	Logical value; TRUE if layer displays object centroids.
LAYER_INFO_SELECTABLE	Logical value; TRUE if the layer is selectable
LAYER_INFO_PATH	String value representing the full directory path of the table associated with the map layer.
LAYER_INFO_TYPE	<p>SmallInt value, indicating this layer's file type:</p> <ul style="list-style-type: none"> • LAYER_INFO_TYPE_NORMAL for a normal layer; • LAYER_INFO_TYPE_COSMETIC for the Cosmetic layer; • LAYER_INFO_TYPE_IMAGE for a raster image layer; • LAYER_INFO_TYPE_THEMATIC for a thematic layer. • LAYER_INFO_TYPE_GRID for a grid image layer.

	<ul style="list-style-type: none"> • LAYER_INFO_TYPE_WMS for a layer from a Web Service Map.
LAYER_HOTLINK_EXPR	Returns the layer's Hotlink filename expression.
LAYER_HOTLINK_MODE	Returns the layer's Hotlink mode, one of the following predefined values: <ul style="list-style-type: none"> • HOTLINK_MODE_LABEL • HOTLINK_MODE_OBJ • HOTLINK_MODE_BOTH
LAYER_HOTLINK_RELATIVE	Returns True if the relative path option is on, False otherwise.

Kullanımı=

LayerInfo(pencere_id, tabakanın_sayısı, tabakadanistenilenbilgi)

Dim pencere_id as integer

Dim tabaka_ad as string

pencere_id=frontwindow()

tabaka_ad= layerinfo(pencere_id, Mapperinfo(pencere_id, Mapper_INFO_LAYERS), LAYER_INFO_NAME)

Yukarıdaki kod bloğunda, görünümde en önde kalan tabakanın ad bilgisi alınmıştır.

Windowinfo()= Var olan pencereler ile ilgili bilgi döndürür.

Kullanımı: *Windowinfo(pencere_id, bilgi)*

pencere_id değeri kullanıcının seçim yaptığı pencerenin numara değeridir.

Frontwindow() fonksiyonu ile elde edilebilir.

Aşağıda fonksiyonun içine eklenecek ikinci parametrede kullanılacak, pencere hakkında elde edilecek bilgi

Bilgi	Geriye döndürdüğü
WIN_INFO_AUTOSCROLL (17)	Logical value: TRUE if the autoscroll feature is on for this window, allowing the user to scroll the window by dragging to the win dow's edge. To turn autoscroll on or off, see Set Window.
WIN_INFO_CLONEWINDOW (15)	String value: a string of MapBasic state ments that can be used in a Run Command statement to duplicate a window. See Run Command .
WIN_INFO_HEIGHT (5)	Float value: window height (in paper units)
WIN_INFO_LEGENDS_MAP (10)	Integer value: when you query a Legend window created using the Create Legend statement, this code returns the Integer win dow ID of the Map or Graph window that owns the legend. When you query the stan dard Legend window, returns 0.
WIN_INFO_NAME (1)	String value: the name of the window.
WIN_INFO_OPEN (11)	Logical value: TRUE if the window is open (used with special windows such as the Info window).

WIN_INFO_SMARTPAN (18)	Logical value; TRUE if Smart Pan has been set on.
WIN_INFO_STATE (9)	SmallInt value: WIN_STATE_NORMAL if at normal size, WIN_STATE_MINIMIZED if minimized, WIN_STATE_MAXIMIZED if maximized.
WIN_INFO_SYSMENUCLOSE (16)	Logical value: FALSE indicates that a Set Window statement has disabled the Close command on the window's system menu.
WIN_INFO_TABLE (10)	String value: For Map windows, the name of the window's "CosmeticN" table. For Layout windows, the name of the window's "Lay outN" table. For Browser or Graph windows, the name of the table displayed in the window.
WIN_INFO_TOPMOST (8)	Logical value: TRUE if this is the active window.
WIN_INFO_TYPE (3)	SmallInt value: window type, such as WIN_LAYOUT. See table below.
WIN_INFO_WIDTH (4)	Float value: window width (in paper units).
WIN_INFO_WINDOWID (13)	Integer value, representing the window's ID; identical to the value returned by WindowID(. This is useful if you pass zero as the window_spec.
WIN_INFO_WND (12)	Integer value. On Windows, the value represents a Windows HWND for the window you are querying.
WIN_INFO_WORKSPACE (14)	String value: the string of MapBasic statements that a Save Workspace operation would write to a workspace to record the settings for this map. Differs from WIN_INFO_CLONEWINDOW in that the results include Open Table statements, etc.
WIN_INFO_X (6)	Float value: the window's distance from the left edge of the MapInfo Professional work area (in paper units).
WIN_INFO_Y (7)	Float value: the window's distance from the top edge of the MapInfo Professional work area (in paper units).
WIN_INFO_PRINTER_NAME (21)	Returns string value with printer identifier (for example, \\DISCOVERY\HP4_DEVEL)
WIN_INFO_PRINTER_ORIENT (22)	Returns WIN_PRINTER_PORTRAIT or WIN_PRINTER_LANDSCAPE
WIN_INFO_PRINTER_COPIES (23)	Returns integer number of copies.
WIN_INFO_SNAPMODE (19)	Returns a logical value. TRUE if snap mode is on. FALSE if snap mode is off.

WIN_INFO_SNAPTHRESHOLD (20)	Returns a SmallInt value representing the pixel tolerance.
WIN_INFO_PRINTER_PAPERSIZE (24)	Integer value. Refer to the Papersize.def file (In the \MapInfo\MapBasic folder) for the meaning of the return value.
WIN_INFO_PRINTER_LEFTMARGIN (25)	Float value: left printer margin value in current units.
WIN_INFO_PRINTER_RIGHTMARGIN (26)	Float value: right printer margin value in current units.
WIN_INFO_PRINTER_TOPMARGIN (27)	Float value: top margin value in current units.
WIN_INFO_PRINTER_BOTTOMMARGIN (28)	Float value: bottom printer margin value in current units.
WIN_INFO_PRINTER_BORDER (29)	String value: ON if a black border will be on the printer output, OFF otherwise.
WIN_INFO_PRINTER_TRUECOLOR (30)	String value: ON if use 24-bit true color to print raster and grid images. This is possible when the image is 24 bit and the printer supports more than 256 colors, OFF otherwise.
WIN_INFO_PRINTER_DITHER (31)	String value: return dithering method, which is used when it is necessary to convert a 24-bit image to 256 colors. Possible return values are HALFTONE and ERRORDIFFUSION. This option is used when printing raster and grid images. Dithering will occur if WIN_INFO_PRINTER_TRUECOLOR is disabled or if the printer color depth is 256 colors or less.

Eğer bilgi kısmında *WIN_INFO_TYPE* kullanılırsa, aşağıdaki tabloda Geri Dönen sütünü içindeki veri, geri dönecektir. Kullanıcının tıkladığı pencerenin harita penceresi, tablo penceresi, çıktı penceresi,... penceresi olduğuna dair geri dönüş olacaktır. *Windowinfo()* fonksiyonun bu geri dönüş bilgisi sayesinde kullanıcının harita penceresine tıklayıp tıklamadığı sorgulanabilir.

Geri Dönen	İçeriği
WIN_MAPPER	Map window
WIN_BROWSER	Browse window
WIN_LAYOUT	Layout window
WIN_GRAPH	Graph window
WIN_HELP	The Help window
WIN_MAPBASIC	The MapBasic window
WIN_MESSAGE	The Message window (used with the MapBasic Print statement)
WIN_RULER	The Ruler window (displays the distances measured by the Ruler tool)
WIN_INFO	The Info window (displays data when the user clicks with the Info tool)
WIN_LEGEND	The Theme Legend window
WIN_STATISTICS	The Statistics window
WIN_MAPINFO	The MapInfo application window

WIN_BUTTONPAD	A ButtonPad window
WIN_TOOLBAR	The Toolbar window
WIN_CART_LEGEND	The Cartographic Legend window
WIN_3DMAP	The 3D Map window

Objectinfo()= Grafik obje hakkında geriye bilgi gönderen fonksiyondur.

Kullanımı= *Objectinfo(obje, bilgi)*

Objectinfo() fonksiyonunun ilk parametresi obje veritipinde bir değişken ya da kayıtlı tabakadaki obje olmalıdır. Eğer *objectinfo()* fonksiyonunun ilk parametresi olarak tabakada kayıtlı objeyse, parametre girişi olarak *tabaka_adi.obj* formatı kullanılmalıdır. İkinci parametre ise objeden istenilen bilgidir.

Bilgi	İçeriği
OBJ_INFO_TYPE (1)	SmallInt, representing the object type; the return value is one of the values listed in the table below (for example, OBJ_TYPE_LINE). This attribute from the DEF file is 1 (ObjectInfo(Object,1)).
OBJ_INFO_PEN (2)	Pen style is returned; this query is only valid for the following object types: Arc, Ellipse, Line, Polyline, Frame, Regions, Rectangle, Rounded Rectangle.
OBJ_INFO_BRUSH (3)	Brush style is returned; this query is only valid for the following object types: Ellipse, Frame, Region, Rectangle, Rounded Rectangle.
OBJ_INFO_TEXTFONT (2)	Font style is returned; this query is only valid for Text objects. Note: If the Text object is contained in a mappable table (as opposed to a Layout window), the Font specifies a point size of zero, and the text height is controlled by the Map window's zoom distance.
OBJ_INFO_SYMBOL (2)	Symbol style; this query is only valid for Point objects.
OBJ_INFO_NPNTS (20)	Integer, indicating the total number of nodes in a polyline or region object.
OBJ_INFO_SMOOTH (4)	Logical, indicating whether the specified Polyline object is smoothed.
OBJ_INFO_FRAMEWIN (4)	Integer, indicating the window id of the window attached to a Frame object.
OBJ_INFO_FRAMEITITLE (6)	String, indicating a Frame object's title.
OBJ_INFO_NPOLYGONS (21)	SmallInt, indicating the number of polygons (in the case of a region) or sections (in the case of a polyline) which make up an object.
OBJ_INFO_NPOLYGONS+N (21)	Integer, indicating the number of nodes in the Nth polygon of a region or the Nth section of a polyline. Note: With region objects, MapInfo Professional counts the starting node twice (once as the start node

	and once as the end node). For example, ObjectInfo returns a value of 4 for a triangle shaped region.
OBJ_INFO_TEXTSTRING (3)	String, representing the body of a Text object; if the object has multiple lines of text, the s
OBJ_INFO_TEXTSPACING (4)	Float value of 1, 1.5, or 2, representing a Text object's line spacing.
OBJ_INFO_TEXTJUSTIFY (5)	SmallInt, representing justification of a Text object: 0 = left, 1 = center, 2 = right.
OBJ_INFO_TEXTARROW (6)	SmallInt, representing the line style associated with a Text object: 0 = no line, 1 = simple line, 2 = arrow line.
OBJ_INFO_FILLFRAME (7)	Logical: TRUE if the object is a frame that contains a Map window, and the frame's "Fill Frame With Map" setting is checked.
OBJ_INFO_NONEMPTY (11)	Logical, returns TRUE if a Multipoint object has nodes, FALSE - if object is empty.
OBJ_INFO_REGION (8)	Object value representing region part of a collection object. If collection object does not have a region, it returns empty region. This query is valid only for collection objects
OBJ_INFO_PLINE (9)	Object value representing polyline part of a collection object. If collection object does not have a polyline, it returns empty polyline object. This query is valid only for collection objects
OBJ_INFO_MPOINT (10)	Object value representing Multipoint part of a collection object. If collection object does not have a Multipoint, it returns empty Multipoint object. This query is valid only for collection objects
OBJ_INFO_Z_UNIT_SET(12)	Logical, indicating whether Z units are defined.
OBJ_INFO_Z_UNIT(13)	String result: indicates distance units used for Z-values. Return empty string if units are not specified.
OBJ_INFO_HAS_Z(14)	Logical, indicating whether object has Z values.
OBJ_INFO_HAS_M(15)	Logical, indicating whether object has M values

Eğer İkinci parametre OBJ_INFO_TYPE kullanılırsa, objectinfo() fonksiyonu objenin hangi tipe obje olduğuyula ilgili olarak, aşağıdaki tabloda Geriye Dönen sütunundaki, bilgiyi döndürür.

Geriy e Döner	İçeriđi
OBJ_TYPE_ARC	Arc object
OBJ_TYPE_ELLIPSE	Ellipse / circle objects
OBJ_TYPE_LINE	Line object
OBJ_TYPE_PLINE	Polyline object
OBJ_TYPE_POINT	Point object
OBJ_TYPE_FRAME	Layout window Frame object
OBJ_TYPE_REGION	Region object
OBJ_TYPE_RECT	Rectangle object
OBJ_TYPE_ROUNDRECT	Rounded rectangle object
OBJ_TYPE_TEXT	Text object
OBJ_TYPE_MULTIPPOINT	Collection text object

SearchInfo()= Seçilen harita objesi hakkında bilgi elde etmek için kullanılır.

Kullanımı= SearchInfo(obje_sayisi, bilgi)

obje_sayisi, haritada seçim yapılan yerde var olan obje sayısı. Eğer bir nokta seçiliyor ise *obje_sayisi* değeri 1 olmalıdır.

bilgi, seçilen obje hakkında bilgi. Aşağıdaki tabloda kullanılacak bilgi ve kullanıldığında geriye dönen değer yazılmıştır.

Bilgi	Geriye Dönen Değer
SEARCH_INFO_TABLE	Objenin bulunduğu tabaka adı.
SEARCH_INFO_ROW	Tabloda kaçınıcı kayıtlı obje olduğuna dair bilgi.

Ek -1Semt açısının Hesaplama adımları:

- İlk olarak semtin hangi noktadan hangi noktaya doğru olduğu belirlendikten sonra, noktaların Y ve X koordinatlarının farkı alınacak. Koordinat farkı alınırken, örneğin semt (P. 1 – P. 2) olarak isteniyorsa, $\Delta Y = Y_2 - Y_1$, $\Delta X = X_2 - X_1$ olarak hesaplanacak. Eğer (P. 2 – P. 1) semti hesaplanması isteniyorsa koordinat farkları hesaplanırken: $\Delta Y = Y_1 - Y_2$, $\Delta X = X_1 - X_2$ olarak hesaplanmalıdır.
- İkinci kısım ise koordinat farklarının pozitif veya negatif olduğuna göre semt açısının hangi bölgede olduğunu belirlemektir. Belirlerken yapılacak olan, X – Y eksenlerini çizip çıkan farkların + veya – olmasına göre hangi eksenler de olduğunu şekil üzerinde göstermek olacaktır.

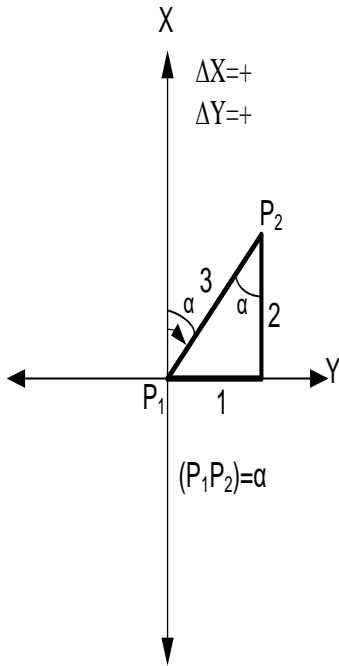
Tablo 2’de koordinat farklarının negatif/pozitif olmasına göre bölgelerin belirlenmesini göstermektedir. Şekil 70, ΔY ve ΔX farklarının pozitif veya negatif olmasına göre bölgelerin basit eksen çizimleriyle nasıl bulunacağını göstermektedir.

Tablo 10 Koordinat farklarına göre bölgelerin belirlenmesi

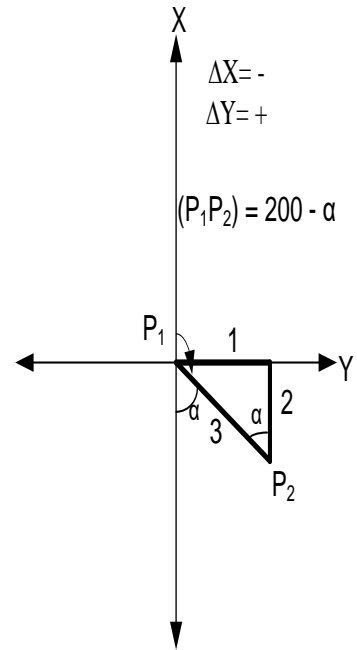
ΔY Farkı	ΔX Farkı	Bölge
+	+	<i>I.Bölge</i>
+	–	<i>II.Bölge</i>
–	–	<i>III.Bölge</i>
–	+	<i>IV.Bölge</i>

Semt açısı her bölgede farklı hesaplanır. Temel formül FORMÜL 1’de gösterildiği gibidir, fakat farklı bölgelerde sadece bu formül doğru sonuç vermeyecektir. Şekil 71 incelendiğinde her bölgede bir α açısı görülmektedir. Bu açı FORMÜL 1’de ki hesapta bulunan açıdır. Hesaplanmasında ΔX ve ΔY değerlerinin mutlakları alınır. α açısı, semt açısını bulmakta yardımcı açı olarak kullanılır.

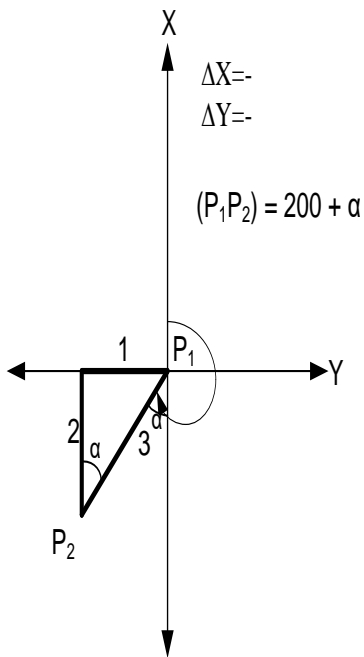
I. Bölge



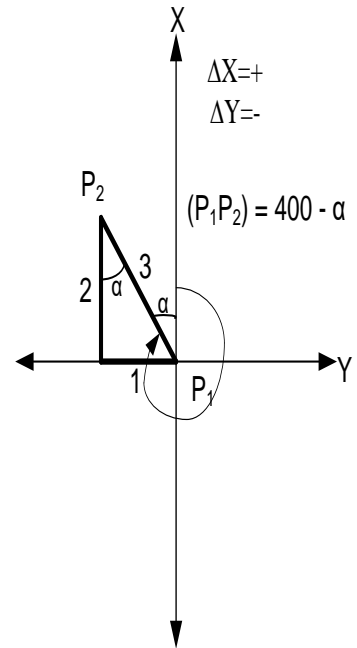
II. Bölge



III. Bölge



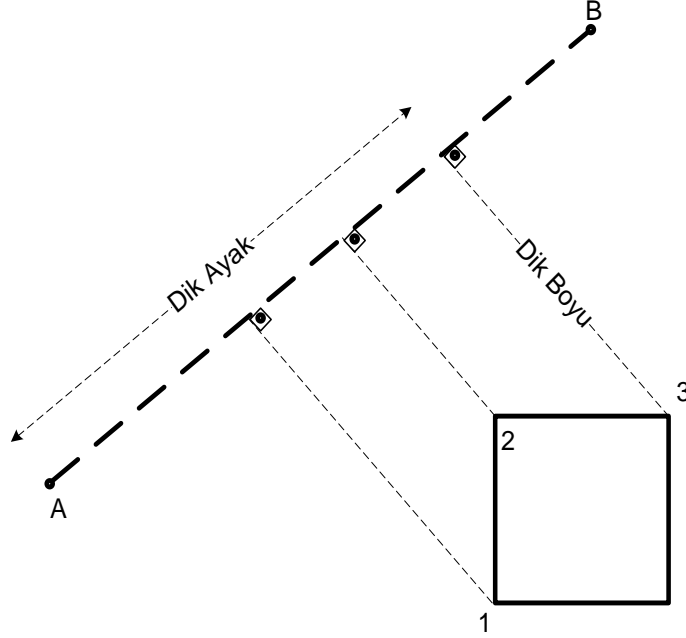
IV. Bölge



CMD_INFO_X

Ek – 2 Yan Nokta Hesabı

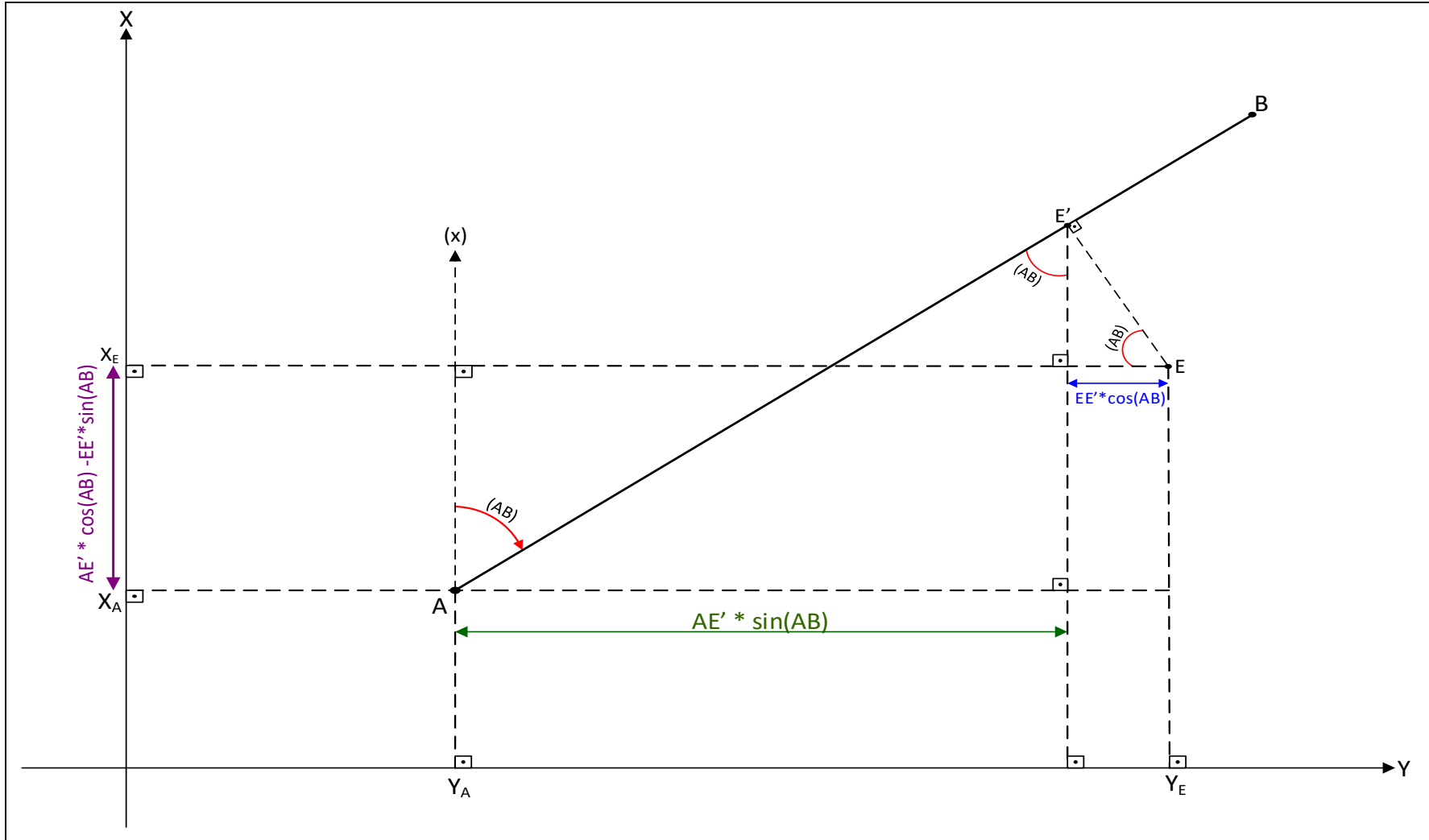
Yan nokta hesabı, koordinatı bilinen iki noktanın oluşturduğu doğrunun sağında ve solunda kalan noktaların koordinatlarının dik boy ve dik ayak uzunlukları yardımıyla bulunmasına denir.



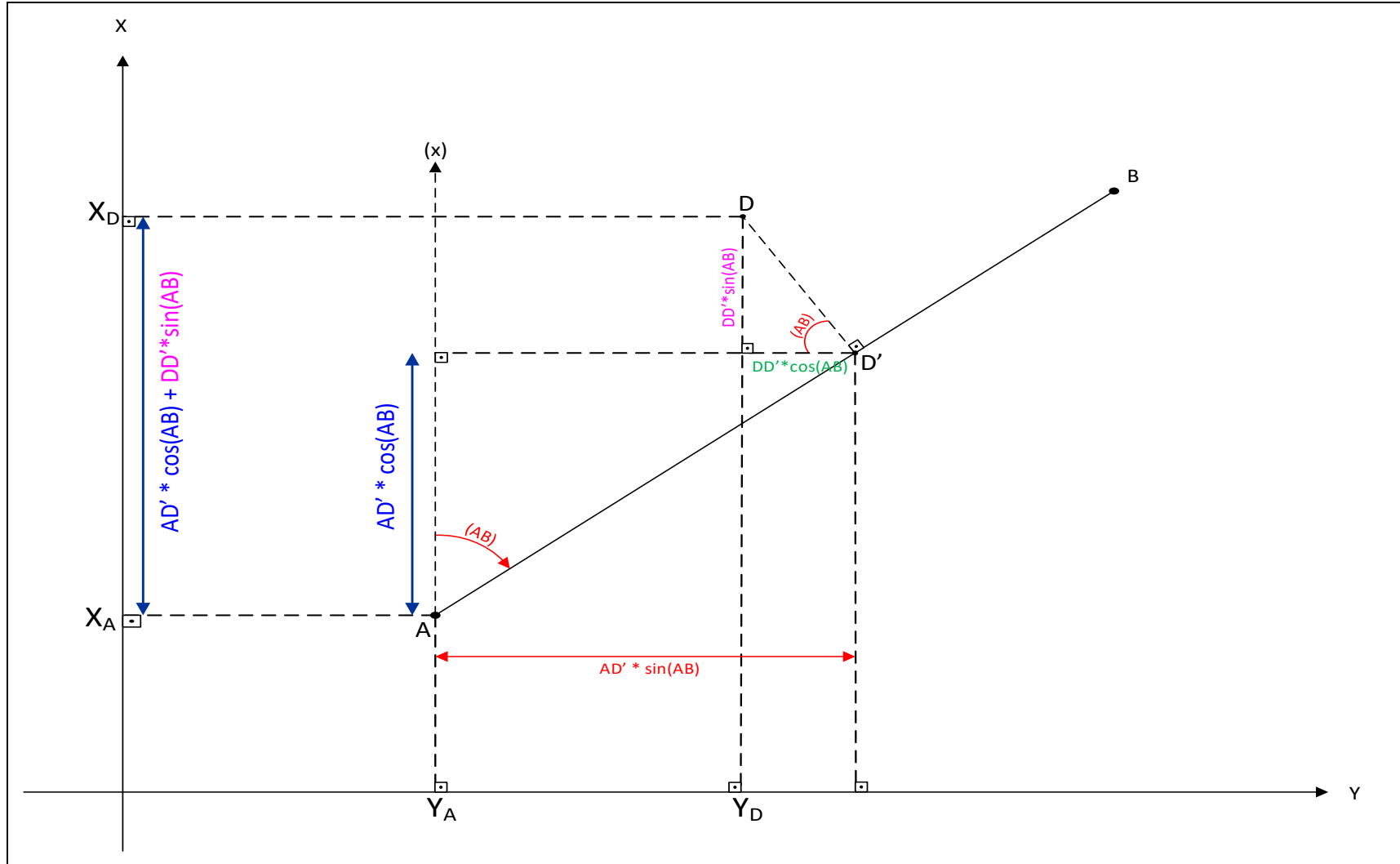
Şekil 163'de yan nokta hesabına bir örnek verilmiştir. A ve B noktaları koordinatları bilinen noktalardır. E noktasının koordinatları bulunmak istendiğinde hangi noktaya göre koordinatlarının hesaplanacağı bilinmelidir. Bir nevi A-B güzergâhının başlangıç noktasının bilinmesi ve koordinatı bulunacak olan nokta güzergâhın solunda veya solunda olmasına benzer bir şekilde E noktasının koordinatı hesaplanacaktır. E noktasının koordinatları:

$$Y_E = Y_A + \overline{AE'} * \sin(AB) + \overline{EE'} * \cos(AB) \rightarrow \overline{AE'} = \text{dik ayak'}$$

$$X_E = X_A + \overline{AE'} * \cos(AB) - \overline{EE'} * \sin(AB) \rightarrow \overline{EE'} = \text{dik boyu}$$



Şekil 163



Şekil 164

Şekil 164'de farklı olarak koordinatı bulunacak olan nokta A-B doğrusunun solundadır.

Değişen formüllerdeki işaret farkı olacaktır.

$$\begin{aligned} Y_D &= Y_A + \overline{AD'} * \sin(AB) - \overline{DD'} * \cos(AB) & \overline{AD'} &= \text{dik ayak}' \\ X_D &= X_A + \overline{AD'} * \cos(AB) + \overline{DD'} * \sin(AB) & \overline{DD'} &= \text{dik byu} \end{aligned}$$

Genel bir formül çıkartıldığında:

*A doğrunun başlangıç noktası, B doğrunun bitiş noktası olduğu ve koordinatı bulunacak olan nokta güzergâhın solunda ise dik boyu değeri (-1) ile çarpılıp işleme konulacak.

Genel formül aşağıda verilmiştir:

$$Y_{Nokta} = Y_{Başlangıçnoktası} + \text{dik ayak} * \sin(AB) + \text{dik boyu} * \cos(AB)$$

$$X_{Nokta} = X_{Başlangıçnoktası} + \text{dik ayak} * \cos(AB) - \text{dikboyu} * \sin(AB)$$



Dik ayak değerleri, A noktasından itibaren ölçülür. Kontrol amacıyla $\overline{A-B}$ uzunluğu ölçülür. Ölçülen $\overline{A-B}_0$ uzunluğu, koordinatlardan hesaplanacak olan $\overline{A-B}_h$ uzunluğuyla karşılaştırılır. Değer aynı çıkmayabilir.

$$d_s = 0.003 * \sqrt{\overline{A-B}_h} + 0.0001$$

$$d = \overline{A-B}_h - \overline{A-B}_0 \quad \rightarrow \quad d_s > d \quad \text{olmalıdır}$$

$d_s > d$ olması durumunda kullanılacak formül:

$$\sin(A - B) = \frac{Y_B - Y_A}{\overline{A-B}_0} = a \quad , \quad \cos(A - B) = \frac{X_B - X_A}{\overline{A-B}_0} = b$$

Genel formül tekrar düzenlenirse:

$$Y_{Nokta} = Y_{Başlangıçnoktası} + \text{dik ayak} * a + \text{dik boyu} * b$$

$$X_{Nokta} = X_{Başlangıçnoktası} + \text{dik ayak} * b - \text{dikboyu} * a$$

