



# 2. Bölüm

**Programlama Temel Bilgisi**

## 2.1. Bilgisayar Sistemleri

### Bilgisayar

Bilgisayar giriş birimleri ile dış dünyadan aldıkları veriler üzerinde, aritmetiksel ve mantıksal işlemler yaparak saklayabilen, sakladığı bilgilere istenildiğinde ulaşılabilen ve çıkış birimleri ile dış dünyaya ileten yazılım ve donanım tabanlı sayısal elektronik bir sistemdir.

### Bellek (Memory)

Bellek veri depolama amacıyla kullanılan birimdir. Her bir bellek hücresinin ayrı bir adresi vardır. Mikroişlemci ve bellek arasındaki veri iletişiminde, bellekten okuma veya yazma amacıyla ilgili bellek hücrelerinin adresleri kullanılır.

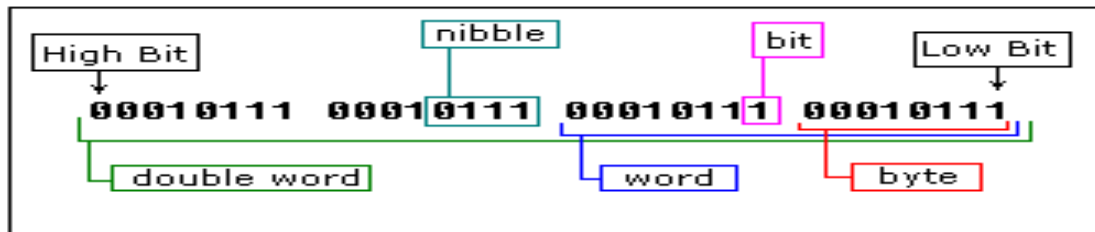
Belleklerin temelinde 0 ve 1 olarak adlandırılan BIT (**B**inary **D**igi**T**) yapıları vardır.

Tablo 2.1’de elektronik ve bilgisayar sistemlerinde kullanılan veri büyüklükleri görülmektedir. Şekil 2.1’de ise veri büyüklüklerin İkili Sayı sisteminde ifade edilmesi görülmektedir.

**Tablo 2.1.** Veri büyüklükleri

Veri Tanımı	Birim	Açıklama
0 veya 1	Bit	Bit
4 adet Bit (1010)	Nibble	Nibble
8 adet Bit (01000001)	Byte	Byte
1024 x Byte	KiloByte	KB
1024 x KByte	MegaByte	MB
1024 x MByte	GigaByte	GB
1024 x GByte	TeraByte	TB
1024 x TByte	PetaByte	PB
1024 x PByte	ExaByte	EB
1024 x EByte	ZettaByte	ZB
1024 x ZByte	YottaByte	YB

**Not:**  $2^{10} = 1024$  olduğundan birimlerarası dönüşümde 1024 kullanılır.  
**Kb:** KiloBit    **KB:** KiloByte    (**B:** Byte ve **b:**Bit)






**Şekil 2.1.** Veri büyüklükleri

### 2.2. Programlama Temelleri

#### Program

**Program**, bir problemin çözümüne ilişkin olarak yapılacak işlemlerin belirli kurallar içerisinde bir araya getirilmesiyle oluşan veri ve komut kodu dizileridir. Şekil 2.2’de çeşitli programlama dillerinde yazılmış toplama işlemine yönelik örnek programlar görülmektedir.

	<pre>using System; public class Test {     public static void Main()     {         int sayi_1, sayi_2, toplam;         Console.WriteLine("Toplama İşlemi");         Console.WriteLine("1. Sayıyı Giriniz:");         sayi_1 = Convert.ToInt32(Console.ReadLine());         Console.WriteLine("2. Sayıyı Giriniz:");         sayi_2 = Convert.ToInt32(Console.ReadLine());         toplam = sayi_1 + sayi_2;         Console.WriteLine("Toplam : " + toplam);         Console.ReadKey();     } }</pre>
	<pre>fprintf("Toplama İşlemi \n ") Sayi_1 = input ('1. Sayıyı Giriniz: '); Sayi_2 = input ('2. Sayıyı Giriniz: '); Toplam = Sayi_1 + Sayi_2; fprintf(" Toplama Sonucu = %d \n", Toplam);</pre>
	<pre>Program Toplama_Islemi; Uses Crt; Var sayi_1, sayi_2, toplam : Integer; Begin     Write ('1. Sayıyı Giriniz:'); Readln(sayi_1);     Write ('2. Sayıyı Giriniz:'); Readln(sayi_2);     toplam := sayi_1 + sayi_2;     Write ('Toplama Sonucu :', toplam);     Write ('Herhangi bir tuşa basınız...');     Repeat Until Keypressed; End.</pre>

Şekil 2.2. Farklı dillerde program örnekleri

## Programlama

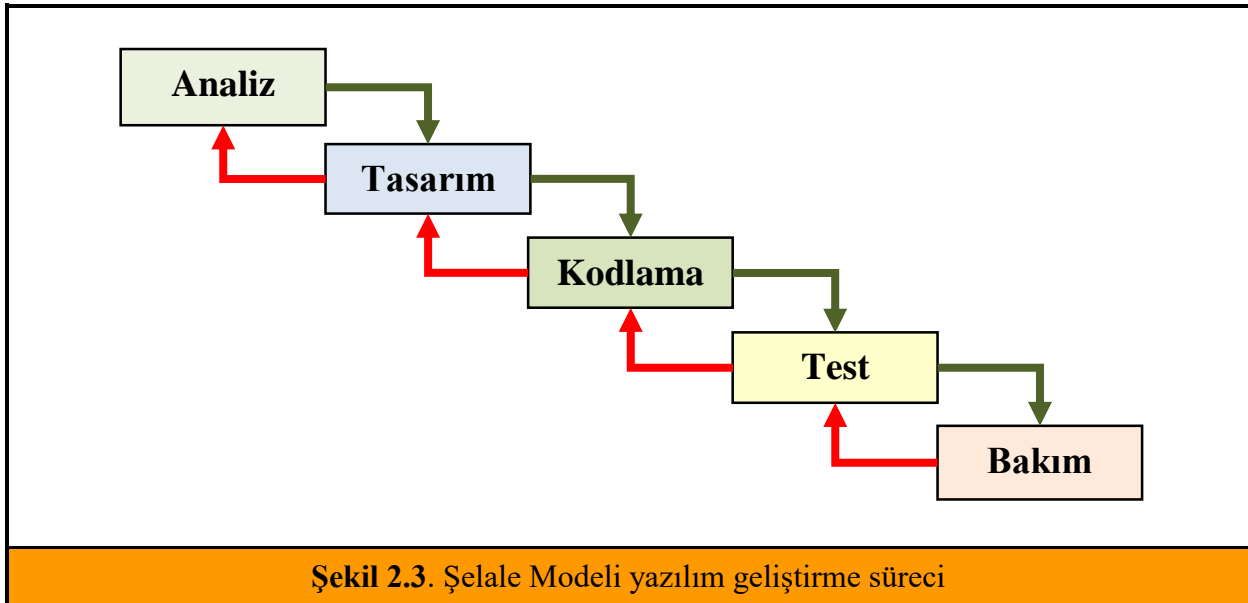
Programlama, programların tasarımı, yazılıp kodlanması, test edilmesi, hataların düzeltilmesi ve gerekli bakımların yapılması sürecidir.

## Yazılım (Software)

Bilgisayar sisteminin çalışması ve işlevlerini yerine getirilebilmesi amacıyla donanım dışında kalan program, veri, kütüphane, doküman vb. tüm sanal sistem yazılım olarak adlandırılır. Yazılım, bilgisayar programlarından oluşmaktadır.

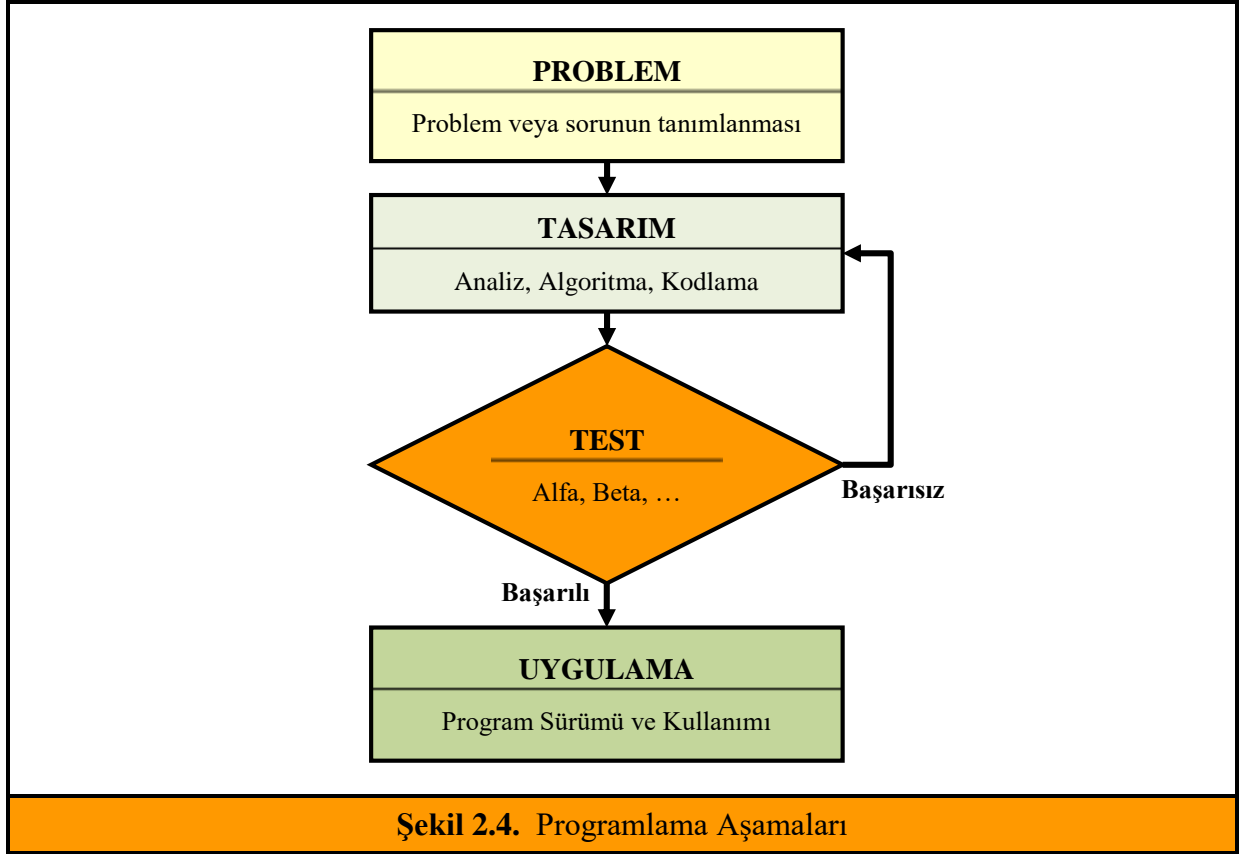
## Yazılım Geliştirme Süreci Modeli (Waterfall – Şelale Modeli)

Yazılım geliştirmede kullanılan klasik süreçlerden biri Şelale Modelidir. Bu modelde yazılım geliştirme süreci; Analiz, Tasarım, Kodlama, Test ve Bakım olmak üzere 5 aşamadan oluşmaktadır (Şekil 2.3).



### 2.3. Programlama Aşamaları

Programlama stratejisi genel anlamda Problem Tanımı, Tasarım, Test ve Uygulama aşamalarından oluşmaktadır (Şekil 2.4).



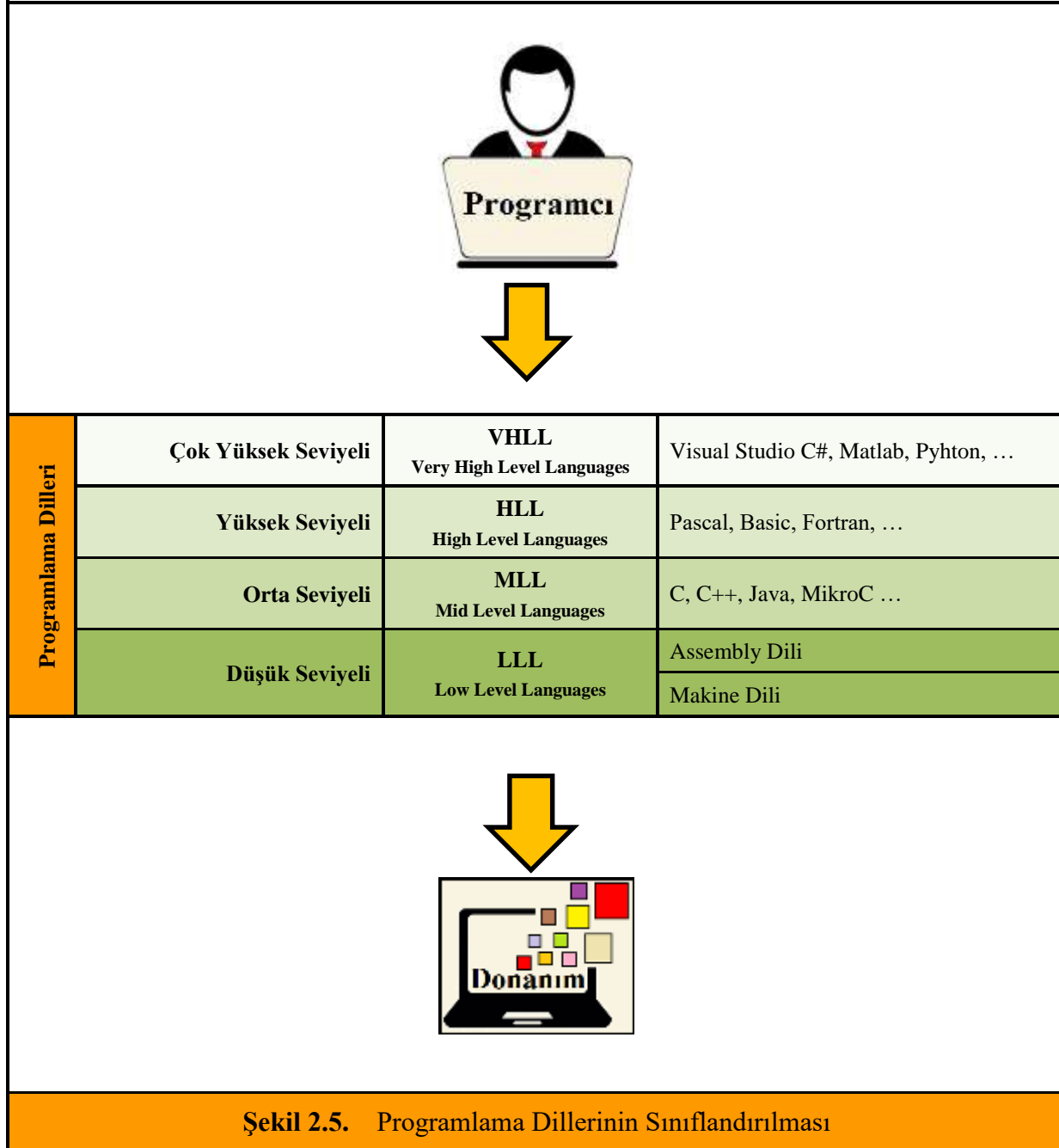
Programlama işleminde, problemin doğru anlaşılması, ihtiyaçların belirlenmesi ve çözüme yönelik analizin iyi bir şekilde yapılması gerekmektedir. Gerekli araştırma ve analiz yapıldıktan sonra, yapılması istenen işin kotarılması veya çözülmesi istenen problemin çözümüne yönelik olarak ayrıntılı algoritmalar hazırlanmalıdır. Algoritma analizi yapılarak, çözüme cevap veren algoritma için uygun bir programlama dilinde kodlama yapılmalıdır.

Programlama stratejisinin en önemli kısmı Test aşamasından oluşmaktadır.

Test aşamasında başarısız olunarak istenen sonuca ulaşamadığı durumlarda, tasarım aşamasının yeniden gözden geçirilmesi ve programlamanın yeniden yapılması gerekmektedir.

## 2.4. Programlama Dilleri

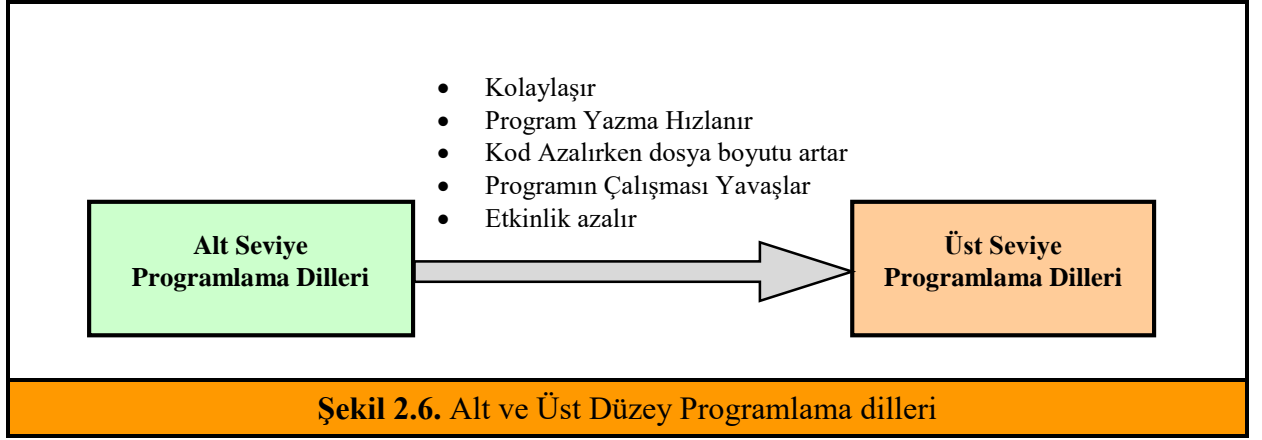
**Programlama Dili**, bir programın yazılması amacıyla daha önceden oluşturulmuş kurallar bütünüdür. Yazılan bir programın bilgisayarın anlayacağı makina diline dönüştürülmesinde **derleyici**, **yorumlayıcı** ve **çeviricilerden** yararlanılmaktadır. Programlama dillerinin seviyelerine göre sınıflandırılması Şekil 2.5'te görülmektedir.



## 2. Bölüm: Programlama Temel Bilgisi

---

Programlama dilleri arasındaki ilişki Şekil 2.6’da ifade edilmiştir.



### Üst Seviyeli Programlama Dilleri

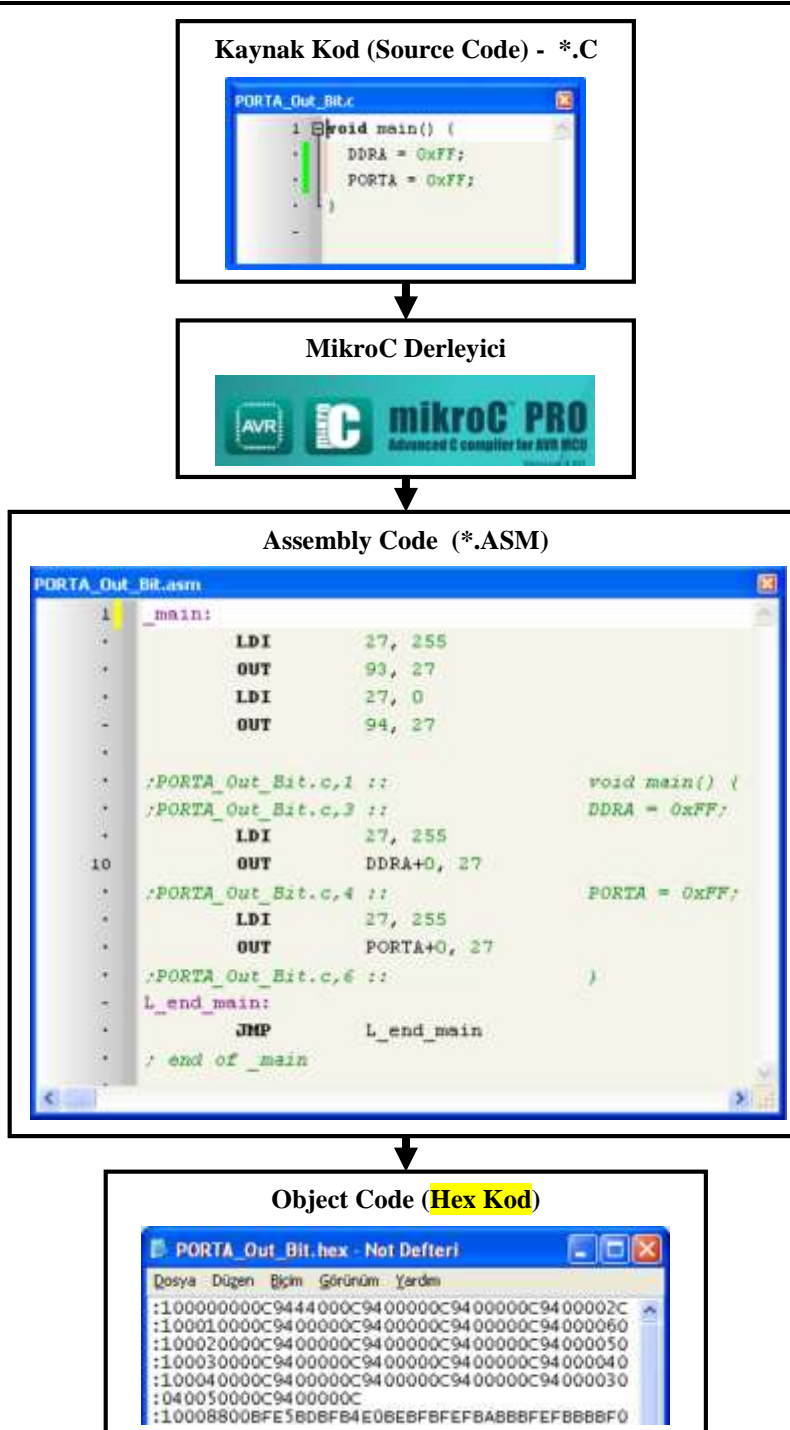
- Text Tabanlı Programlama Dilleri (C# Console, Pascal, GW Basic vb.)
- Görsel Programlama Dilleri (Visual Studio C#, Delphi, Visual Basic vb.)
- Grafik Programlama Dilleri (Labview, WorkBench, Parsic vb.)

Genel olarak programlama dilleri, Program Yazım Kurallarına ilave olarak aşağıdaki temel konuları içermektedir:

- Sabitler, Veri Tipleri ve Değişkenler
- Operatörler
- Karar ve Çevrim Kontrol Yapıları
- Altprogramlar

## 2.5. Programların İşletilmesi

Bilgisayar programların işletilebilmesi için öncelikle yazılan kaynak kodun (Source Code), yorumlayıcı, derleyici, çevirici vb. araçlarla bilgisayarın anlayacağı dile dönüştürülmesi gerekmektedir (Şekil 2.7).



Şekil 2.7. MikroC ile HEX kodların elde edilmesi aşamaları



## 2. Bölüm: Programlama Temel Bilgisi

### Yorumlayıcı (Interpreter)

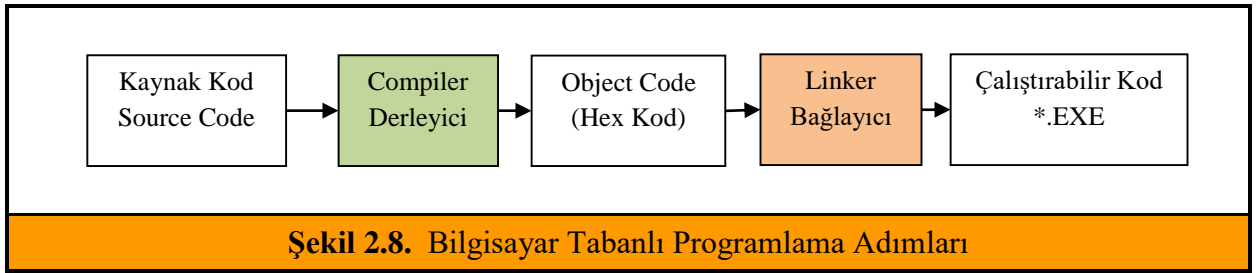
**Yorumlayıcı**, yazılan kaynak programı deyim deyim CPU'nun anlayacağı makine diline dönüştüren ve çalıştıran yazılımlardır.

Programı oluşturan komut satırları birer birer ele alınır. Komut satırında hata yoksa çalıştırılır ve bir sonraki komut satırına geçilir. Hata olduğu anda program çalışması durur. Bu işlem bu şekilde program sonuna kadar devam eder. Yorumlayıcı mantığında programın tamamının hatalardan arındırılmış olması gerekmez. Hatta hataya rastlanılmadığı, şartlar hatalı satırın olduğu komut satırının çalıştırılmasını gerektirmediği sürece program çalışmasına devam eder.

### Derleyici (Compiler)

**Derleyici**, yazılan kaynak programı CPU'nun anlayacağı makine diline dönüştüren yazılımlardır (Şekil 2.8).

Programın tamamı gözden geçirilir ve hiç hata yoksa program çalıştırılabilir hale gelir. Hatalar varsa; hata listesi verilir.



### Bağlayıcı (Linker)

**Bağlayıcı**, derleme ile elde edilen Object kodların ilave durumdaki kütüphane vb. ekstra kodlar ile birlikte kendi başına çalıştırılabilecek hale getiren yazılımlardır.

### Çevirici (Assembler)

Assembly dilinde Mnemonik komutlardan oluşan ve text tabanlı olarak yazılmış programın HEX kodlara dönüştürülmesini sağlayan yazılımlara Assembler-Çevirici adı verilir (Şekil 2.9).

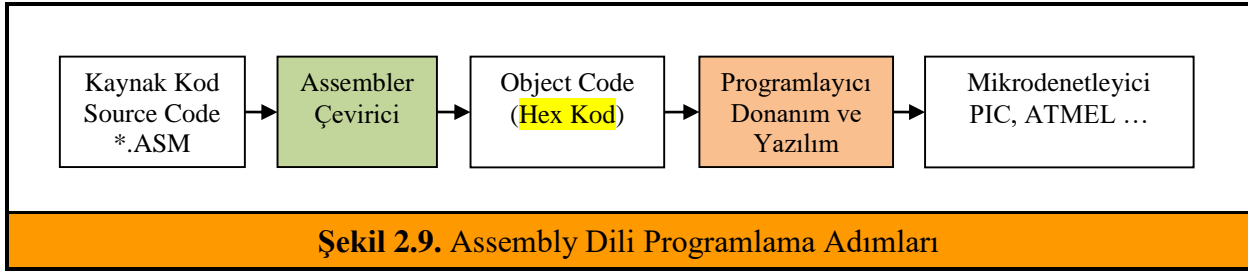
#### Mnemonik Komut Örnekleri

(MOVLW, ADDF, GOTO, CALL BNC vb. kısaltmalar)

SHL – Shift Left – Sola Kaydır

SHR – Shift Right – Sağa Kaydır

ROR –Rotate Right



## Kullanıcı Arayüzü (User Interface)

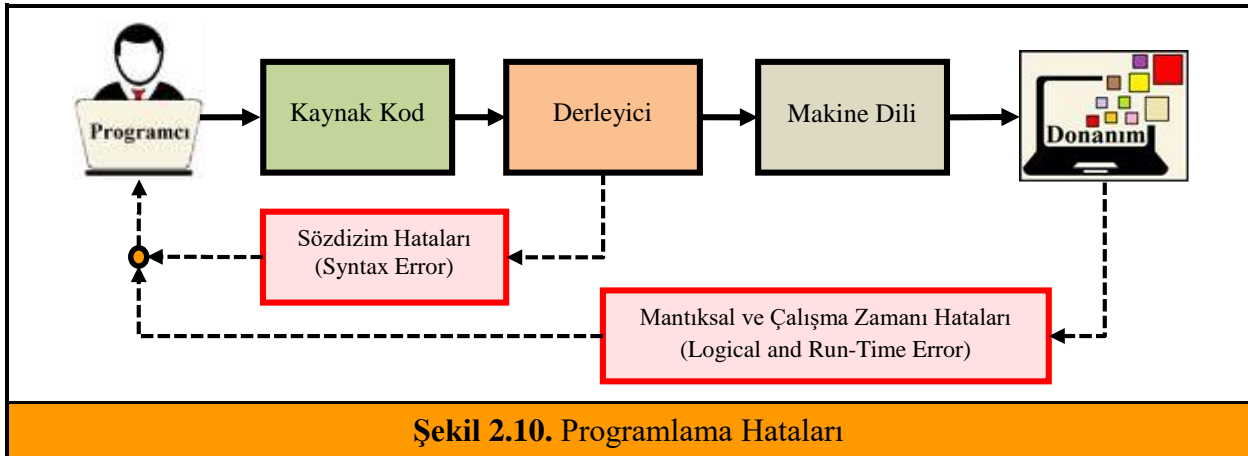
İnsan-Bilgisayar arasında yer alarak etkileşim sağlayan ve veri giriş-çıkış işlemlerinin yönetildiği yazılım arabirimidir. Kullanıcı arayüzleri genel olarak şu şekilde sınıflandırılabilir:

- Text Tabanlı Kullanıcı Arayüzü (TUI - Text-Based User Interface)
- Grafik Tabanlı Kullanıcı Arayüzü (GUI – Graphics User Interface)
- Ses Kullanıcı Arayüzü ( VUI – Voice User Interface)
- Diğer ...

## Programlama Hataları

Program hataları genel olarak 3 sınıfta incelenir (Şekil 2.10):

- Syntax Error (Yazım – Sözdizim Hatası)
- Logical Error (Mantık Hatası)
- Run-Time Error (Çalışma Zamanı Hatası)



**Mantık Hatası**, programın derlenip çalıştırılabilmesine rağmen istenmeyen değerler de üretebilmesi durumudur. Hata program çalışmasında sürekli olarak oluşabileceği gibi, nadiren de meydana gelebilir. Bu açıdan hatanın giderilmesi bazen oldukça zor olabilmektedir.

Programın çalışması esnasında meydana gelebilecek hataları önceden kestirmek ve program akışını bu duruma göre kontrol etmek amacıyla, programlama işlemlerinde **Hata Yakalama Yordamları** sıkça kullanılmaktadır. Örnek Hata Yakalama Yordamları;

Pascal / Delphi : Try/Except/End ve Try/Finally/End  
C# : try / catch / finally

### 2.6. Programlama Önerileri

---

Programın sözdizim, mantık ve çalışma zamanı hatalarından uzak bir şekilde yazılarak doğru çalışması ve doğru sonuçlar üretmesi kodlama ve yazılım geliştirme açısından önemlidir. Programlamada, yazılan bir programın programcının kendisi kadar, başka programcılar tarafından da anlaşılabilir olması arzu edilir.

Bu nedenlerle, iyi bir programda aranan özellikler şu şekilde sıralanabilir:

- Program içinde açıklamalara, kullanım talimatlarına yer verilmelidir.
- İsimlendirmeler kurallara göre ve anlaşılır olarak yapılmalıdır (Deve ve Altçizgi Notasyonu).
- Programda blok yapılarına ve girintilere dikkat edilmelidir.
- Programın anlaşılmasını zorlaştıran break, continue, goto vb. yapısal olmayan kodlamalardan kaçınılmalıdır.

### Kullanıcı Dostu (User Friendly)

Göze hitap eden, daha anlaşılır, daha hızlı sonuca ulaşan, işlemleri kolaylaştıran ve kullanım kolaylığı sağlayan yazılımlar için kullanıcı dostu tabiri kullanılır. Kullanıcı Dostu tabiri bir bakıma yazılımın kalitesini ve kullanılabilirliğini göstermektedir.

## 2.7. IDE – Integrated Development Environment (Tümleşik Geliştirme Ortamı)

IDE (Tümleşik Geliştirme Ortamı), bilgisayar programcılarının hızlı ve rahat bir şekilde yazılım geliştirebilmesini amaçlayan, geliştirme sürecini organize edebilen birçok araç ile birlikte geliştirme sürecinin verimli kullanılmasına katkıda bulunan araçların tamamını içerisinde barındıran bir yazılım türüdür (**wikipedi**).

Yazılım geliştirmede tercih edilen en yaygın IDE araçları Tablo 2.2’de verilmiştir.

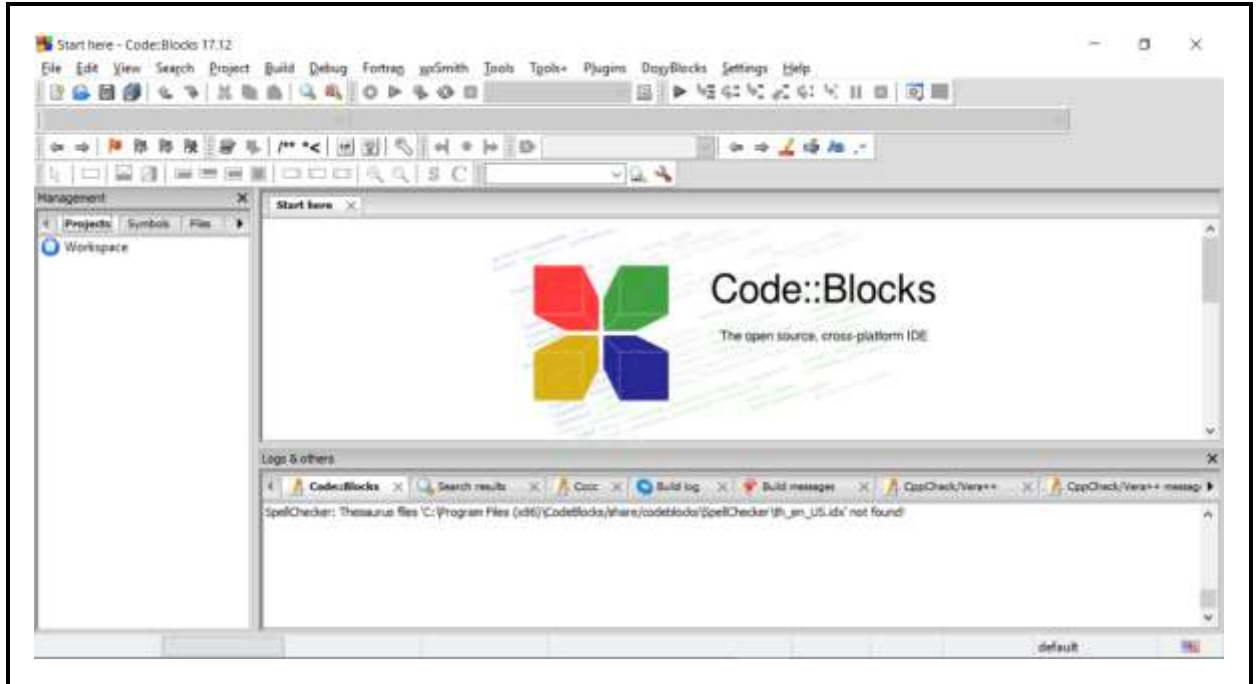
Tablo 2.2. Çeşitli IDE araçları	
IDE Türü	Örnek
Text Tabanlı IDE	C++
Görsel Tabanlı IDE	Microsoft Visual Studio, Matlab
Grafik Tabanlı IDE	Labview
Mobil IDE	Android Studio
Veritabanı IDE	MSSQL, Navicat
Gömülü Sistemler (Embedded Systems)	MikroC Pro for AVR, Arduino IDE

Bir IDE aracında olması gerekli temel özellikler şu şekilde sıralanabilir:

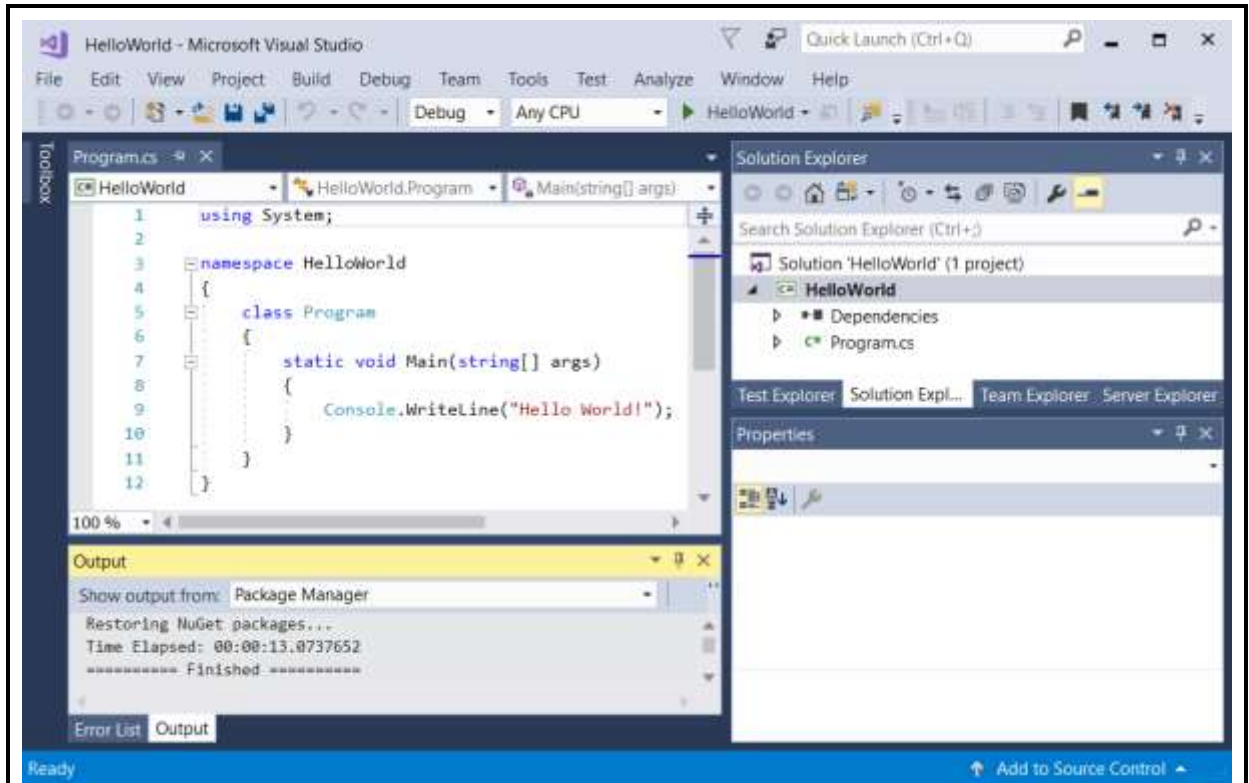
- Kaynak Kod Yazım Editörü
- Kaynak Kod Hiyerarşik Dosya Ağacı
- Tümleşik Derleyici (Compiler), Yorumlayıcı (Interpreter), Builder
- Hata Ayıklayıcı (Debugger)
- Araçlar (Tools)

Şekil 2.11, Şekil 2.12 ve Şekil 2.13’te çeşitli IDE ortamları görülmektedir.

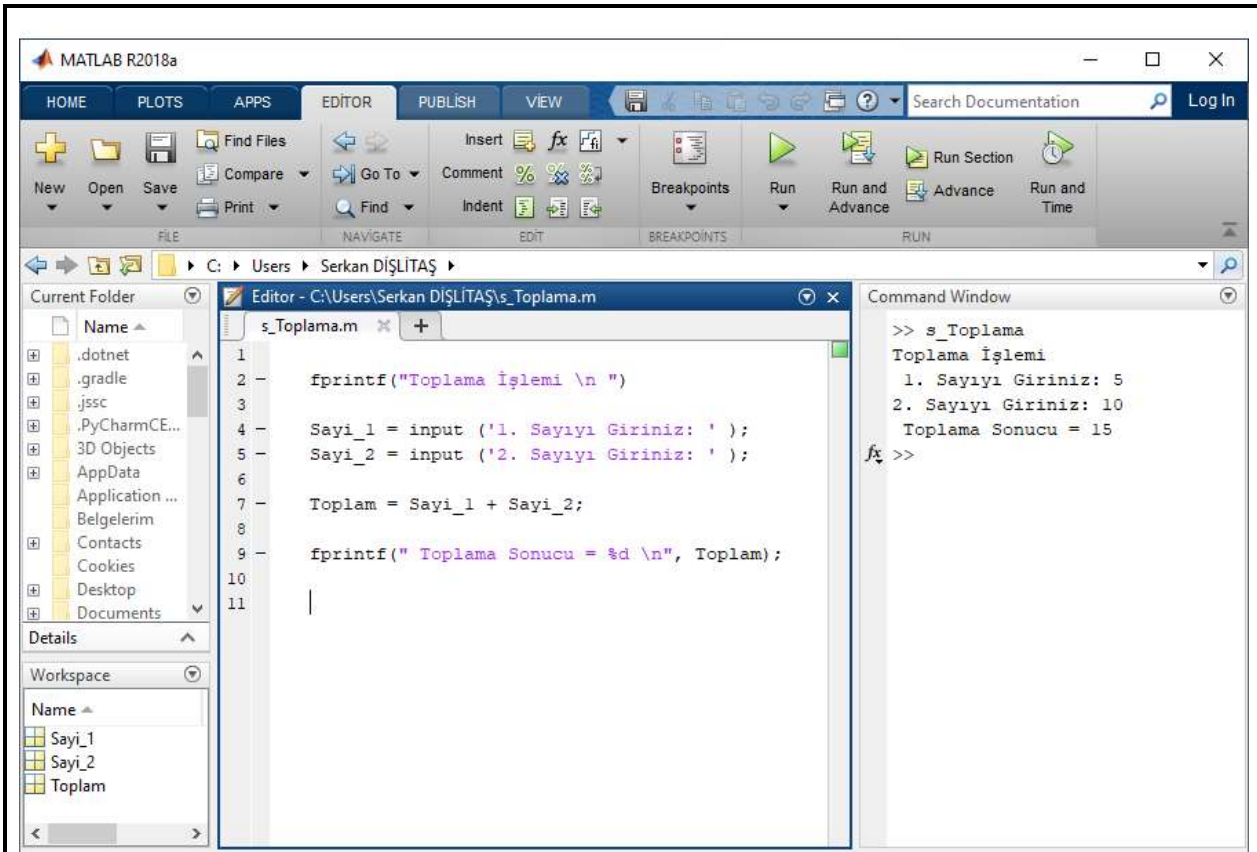
## 2. Bölüm: Programlama Temel Bilgisi



Şekil 2.11. Code::Blocks IDE örnek ekran alıntısı (<http://www.codeblocks.org> )



Şekil 2.12. Visual Studio C# Console IDE örnek ekran alıntısı



Şekil 2.13. MATLAB IDE örnek ekran alıntısı