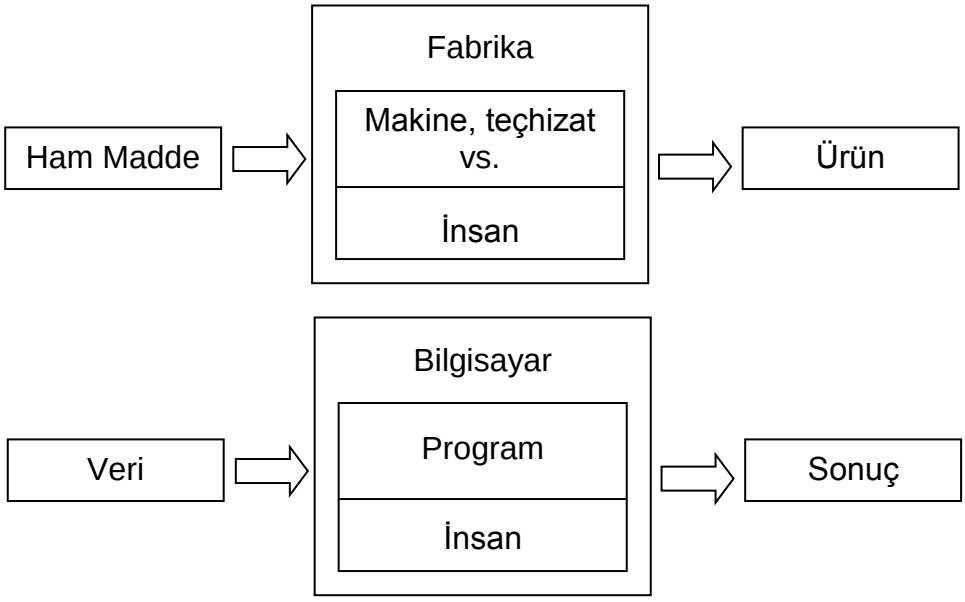
BÖLÜM 1: PROGRAMLAMAYA GİRİŞ

Bilgisayar, verileri klavye, dosya ve veritabanı gibi değişik ortamlardan girdi olarak alan, istendiğinde bunları saklayan, veriler üzerinde çok hızlı işlem yapan ve sonuçları istenen formatta ekran, yazıcı, dosya ve veritabanı gibi ortamlara aktaran bir aygıttır. Bilgisayarlar donanım (hardware) ve yazılım (software) olarak ikiye ayrılır. Donanım bilgisayarı oluşturan fiziksel bileşenlerdir. Yazılım ise donanımı oluşturan bileşenlerin çalışmasını ve gerekli işlevlerini yerine getirmesini sağlayan programlardır.

Bilgisayarlar, insanların yapabileceğinden milyonlarca hatta milyarlarca kat hızlı hesaplama yapabilir ve/ya mantıksal kararlar verilebilirler. İşlenecek veriler ve kullanılacak çözüm yöntemleri, bilgisayara program ile bildirilir – bilgisayarlar problemleri çözebilmek için *"program*"a ihtiyaç duyarlar.

Program ise giriş değerlerini kullanarak istenilen çıkış değerlerinin elde edilebilmesi için bilgisayara iletilen komutlar dizisi veya yapılmasını istediğiniz işlemleri bilgisayara bildirdiğimiz adım adım komutlar kümesidir. Benzer durum bir fabrikada üretimin olabilmesi için hammaddeye ve hammaddeyi işlemek için makine ve teçhizata duyulan ihtiyaç gibidir.



İnsan bilgisayarlara ancak programlarla hükmedebilir. Bilgisayarlar, bilgisayar programı denen komut setlerinin kontrolü altında verileri işlerler. Bilgisayarda çalışan bilgisayar programlarına "yazılım – software" da denir. Bunları geliştiren, yazan kişilere de "bilgisayar programcısı – computer programmer" denir.

I. PROGRAMLAMA DİLLERİ

Programlama dili programcı ile bilgisayar arasındaki iletişimi sağlayan bir araç olup programların yazımında kullanılan bir notasyondur. Bilgisayara istediğiniz işlemleri bildirme yolunu sağlayan kurallar kümesi bir programlama dilini oluşturur. Günümüzde yüzlerle ifade edilen sayıda programlama dili vardır. Bunların birçoğu sadece bilimsel amaçlar için geliştirilmiş olup, yaygın olarak kullanılmamaktadır. Her dilin kendine özgü yazım (syntax) kuralları vardır. Aşağıdaki tablolarda üç farklı dildeki "yaz" ve "gir" komutlarının karşılıkları verilmiştir.

"Yaz" komutu		
Basic	***Pascal***	***C/C++***
Print	Writeln	Printf
"Gir" komutu		
Basic	***Pascal***	***C/C++***
Input	Readln	Scanf

Programlama dillerini programlamanın evrimi bakımından 5 sınıfa ayırabiliriz:

1. Makine Dilleri (1. Kuşak): Veri ve program komutları, bilgisayarın elektriksel devrelerinin açık veya kapalı durumuna karşılık gelen 1 ve 0'larla ifade edilir. Her bilgisayar türünün kendi makine dili vardır. Kodlamanın zor olduğu, hata yapmaya çok açık dillerdir. Bu durum daha kolay yazılabilir ve anlaşılabilir olan birleştirici (assembly) dilinin geliştirilmesini sağlamıştır.

2. Birleştirici (Assembly) Diller (2. Kuşak): Günümüzde bu dil çok düşük seviyeli olarak tanımlanır. Assembly dili ile yazılmış bir program, bir çevirici (translator veya assembler) ile makine diline çevrilerek çalıştırılır. Oldukça anlaşılabilir gözükmesine karşın çok basit bir iş için bile çok sayıda komut yazmak gerekmektedir. Kullanımı çok daha kolay olan dillerin ortaya çıkmış olması nedeniyle, neredeyse hiç kimse artık assembly ile program kodlamamaktadır. Ancak unutulmamalıdır ki bu dil ortaya çıktığında ileriye atılmış çok büyük bir adımdı.

3. Yüksek Seviyeli Diller (3. Kuşak): 1960'larda yaygınlaşmıştır. İngilizceye benzer ifadelerin kullanıldığı, aritmetik işlemlerde kullandığımız işaretlerin kullanıldığı, daha hızlı ve kolay program yazmayı sağlayan, programcının daha az çaba sarf ettiği dillerdir. Çok sayıda dil geliştirilmiştir: Basic, Pascal, Fortran, Cobol, Algol, PLI, C, C++, vb. Bu dillerin ifadeleri derleyici (Compiler) veya yorumlayıcı (Interpreter) tarafından makine diline çevrilirler. Bu kuşak dillerin birçoğuna yordamsal "procedural" diller de denmektedir. Bunun nedeni bu dillerle çözümün "NASIL" yapılacağını ayrıntılı olarak bildirmemizdir.

4. 4. Kuşak Diller (4GL – Fourth Generation Languages): 3. kuşak dillerde yüzlerce satır halinde kodlanan bir program, 4. kuşak dillerle birkaç satırda ifade edilebilir. Çünkü bu dillerle sadece "NE" istediğimizi bildiririz, 3. kuşak dillerde olduğu gibi "NASIL" yapılacağını söylemeyiz. İşin NASIL yapılacağını dil kendi yapısı içerisinde barındırır. Diğer bir deyişle bu diller **"**non-procedural" dir. 4GL diller işlevlerine göre çeşitli gruplara ayrılabilir:

   - Veritabanı Sorgulama Dilleri. Örnek olarak SQL (Structured Query Language). SQL ilişkisel (relational) veritabanı sistemlerinde yaygın olarak kullanılan sorgulama dilidir.
   - Hesap tablosu yazılımları. Örnek: Excel, Lotus, vb.
   - Karar destek sistemleri (Decision Support Systems).
   - İstatistik programları. Örnek: SPSS.
   - Benzetim (Simulation) programları.
   - En iyileme (optimization) programları
   - Grafiksel sunum programları. Örnek: MS Powerpoint, Lotus Smartsuite.

5. 5. Kuşak Diller: Amaç tamamen doğal dillerin özelliklerini taşıyan programlama dilleri üretmektir. Örnek PROLOG ve LISP. Ayrıca, yapay zekâ (artificial intelligence) çalışmalarının önemli bir bölümü bu konuya odaklanmıştır.

Yüksek seviyeli programlama dillerinden bazıları:

Fortran (1954): IBM tarafından geliştirilmiştir. Daha sonra Fortran II, Fortran IV, Fortran 77, Fortran 90, Fortran 95 ve Fortran 2003 sürümleri hazırlanmıştır. Mühendislik ve bilimsel uygulamalar için tasarlanmış bir dildir. Matematiksel formüllerin ifade edilmesi son derece kolaydır. Mühendislik uygulamalarında hala yaygın olarak kullanılır.

COBOL (1954): Komutları İngilizceye yakın. Ticari uygulamalar için uygundur. Günümüzde Cobol ile yazılan birçok ticari uygulama halen kullanılmaktadır.

Basic (1965): Akademik ortam için, Dartmouth Kolejinden Kemeny ve Kurtz tarafından geliştirilmiştir. İlk yıllarda derleyicisi bedava olduğundan çok yaygın olarak ticari alanda, bilimsel alanda ve oyun yazımında kullanılmıştır.

Pascal (1968): İsviçreli Niclaus Wirth tarafından geliştirilmiştir. En çok kullanılan sürümü Borland firmasının Turbo Pascal'ıdır. İş ve bilim dünyasında yıllarca kullanılmıştır.

C (1970): Bell Laboratuvarlarında geliştirilmiştir. Makine diline çevrilmesi kolay ve sistem yazılımları yazmaya çok uygun bir dildir. UNIX işletim sistemi C diliyle yazılmıştır.

C++ (1980): AT&T firmasından Bjarne Stroustrup tarafından geliştirilmiştir. C++, C'ye birtakım ek özellikler ile nesne-yönelimli programlama (object-oriented programming) yetenekleri getirmiştir. Nesneler tekrar tekrar kullanılabilen yazılım bileşenleri olup yazılım geliştirmede büyük kolaylık ve verimlilik sağlar.

ANSI C (1983-1989): C programlama dilinin standartlaştırılmış biçimi olup, 1990'da yayımlanmıştır: ANSI/ISO 9899:1990

Diğer nesne-yönelimli diller:

Smalltalk: Nesne-yönemli (object-oriented) ilk programlama dili olup, Xerox's Palo Alto Araştırma Merkezinde geliştirilmiştir.

Java: 1993'de İnternetin yaygınlaşması ve popülerliği nedeniyle, Sun Microsystems'den James Goslin tarafından 1995 yılında Web sayfası oluşturmak amacıyla geliştirilmiştir. İşletim sisteminden bağımsızdır.

C#: 2000 yılının Haziran ayında Microsoft tarafından kullanıma sunuldu. C#, C ve C++ dillerinden geliştirilmiş ve nesneye yönelik programlama anlayışı üzerine kurulmuş bir dildir. C#, Microsoft tarafından geliştirilmiş .NET platformunun bir parçasıdır. Microsoft'un C# dışında bu platformda çalışan ve nesneye yönelik programlama anlayışı üzerine kurulmuş Visual C++, Visual Basic ve Visulal J# dilleri de mevcuttur.

Delphi (Object Pascal) ve Power Builder'de nesne-yönelimli dillerdendir.

II. PROGRAMLAMA SÜRECİ

Bir program geliştirmenin adımları herhangi bir problem çözme işinin adımlarına çok benzer. Öncelikle problem tanımlanmalı ve sonra çözüm planlanmalıdır. Daha sonra programın işe uygun bir dille kodlanması, sınanması ve belgelenmesi gerekir.

1. Problemin tanımlanması: Bir problemi çözecek programı yazmaya başlamadan önce, problemi tam olarak anlamak ve çözüm için dikkatlice planlama yapmak gerekir. Daha sonra program yazmaya başlanabilir. Programı yazarken de – planlama kadar önemli bir konu - dilin sunduğu yapı bloklarını ve program yapılandırma ilkelerini çok iyi bilmektir. Programı talep eden ve kullanacak olan kişilerle görüşerek, NE istediklerinin, giriş ve çıkış aygıtlarının, istedikleri rapor formatlarının belirlenmesi ve bunlar üzerinde yazılı anlaşmaya varılması gerekir.

2. Çözümün planlanması: Çözümün planlanmasında izlenecek adımlar algoritmalar yardımıyla veya akış diyagramları (flowchart) çizilerek veya pseudocode (sözde kod) yardımıyla belirlenir. Pseudocode (sözde kod) algoritma geliştirmede programcılara yardımcı olan dildir; günlük konuşma dilimizde kullandığımız sözcüklerden oluşur. Bu dille yazılan program bilgisayarda yürütülemez; ancak programcıya geliştirmeye çalıştığı program üzerinde düşünmesini ve onu şekillendirmesini sağlar. İyi yazılmış bir pseudocode'u gerçek bir programlama diline çevirmek kolaydır.

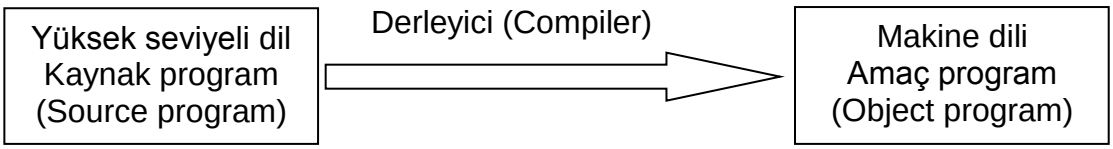
3. Programın kodlanması: Çözümün bir programlama diliyle ifade edilmesi akış şeması ve/veya pseudocode ile ifade ettiğiniz mantığın (algoritmanın) programlama diline tercüme edilme işidir. Kullanacağınız dilin tüm kurallarını doğru olarak takip etmeniz gerekir. Hiçbir yazım hatası yapmadan programınızı kodlamanız, programın hatasız yani istediğiniz sonucu verecek biçimde çalışmasını gerektirmez. Çünkü dili doğru kullanmanıza rağmen mantık hataları (logic errors) yapmış olabilirsiniz. Bunlar, denklem kodlamaları, hatalı işlem (operatör) kullanımları gibi nedenlerden dolayı oluşan hatalardır. Ancak, dilin kurallarının doğru kullanımı gerekli olan ilk adımdır.

4. Programın sınanması: Genellikle, program ne kadar dikkatli kodlanırsa kodlansın, ilk çalıştırmada bir takım derleme hataları (compilation errors - syntax errors) alınması kaçınılmaz olabilir. Bunlar, kullanılan programlama dilinin kurallarına uymayan kodlamalardan dolayı oluşan hatalardır. Derleme aşaması hatasız geçilmeden çalıştırma (run – execute) aşamasına geçilemez. Alınması olası hataları mümkün olduğunca azaltmak ya da yok etmek için programımızı kodladıktan sonra kontrol etmeliyiz. Bunlar:

   a) Masa başı kontroller: Yazdığımız kodun masa başında kontrolü programda kullanılan değişkenlere örnek değerler verilerek koşul cümlelerinin, döngülerin vb. sınanması biçiminde yapılır.

   b) Programın derlenmesi (compilation): Programcı ne kadar deneyimli olursa olsun bu adımda genellikle hatalar alınabilir. Bunları anlayıp düzeltmek gereklidir. Bu hatalar programın derlenmesi sırasında programcıya bildirilir.

Derleyici (Compiler), yüksek seviyeli bir dilde yazılmış olan programı, makine diline çeviren yazılım (program)'dır.



c) Programın çalıştırılması (run – execute) aşaması: Derleme hatalarından arınmış programı çalıştırıp istediğimiz/beklediğimiz sonuçları alıp almadığımızın kontrolü yapılmalıdır. Yani programın mantığı doğru mu? Giriş değeri olarak bir takım örnek değerler verip programı çalıştırdıktan sonra değişik giriş değerleri vererek olabildiğince her koşulda programın çalışıp çalışmadığının kontrol edilmesi önemlidir.

5. Programın belgelenmesi (documentation): Genellikle ihmal edilen ancak **ÇOK ÖNEMLİ** bir aşamadır. Programınızın doğru olarak çalıştığından emin olduktan sonra;

   - programın amacını,
   - nasıl çalıştığını / mantığını, akış şemasını,
   - kullanıcı tarafından nasıl kullanılacağını,
   - giriş bilgilerini (nasıl nereden verilecek, vb.),
   - çıkış bilgilerini, raporlama formatlarını ifade eden, ayrıca
   - program listesini ve örnek çıktıları içeren bir doküman hazırlanmalıdır.

   Bu doküman, programı talep etmiş kullanıcı grubuna verileceği gibi, ilerde değişiklik yapılacağı zaman (sizin tarafınızdan veya başka bir programcı tarafından) çok yardımcı olacaktır. Programcılar en çok, hiçbir dokümantasyonu olmayan mevcut bir programı değiştirmeleri istendiğinde zorlanırlar.

III. C PROGRAMLAMA DİLİ

C Programlama Dili daha önce de belirttiğimiz gibi genel amaçlı yapısal bir programlama dilidir. 1972 yılında Dennis Ritchie tarafından Bell Telefon Laboratuvarında tasarlanmıştır. C, özellikle sistem programlamada sembolik makine dili (assembler) ile tercih edilmektedir. İşletim sistemleri, derleyiciler ve debug programları gibi düşük seviyeli sistem programlarının yazılımında yoğun olarak C programlama dili kullanılır.

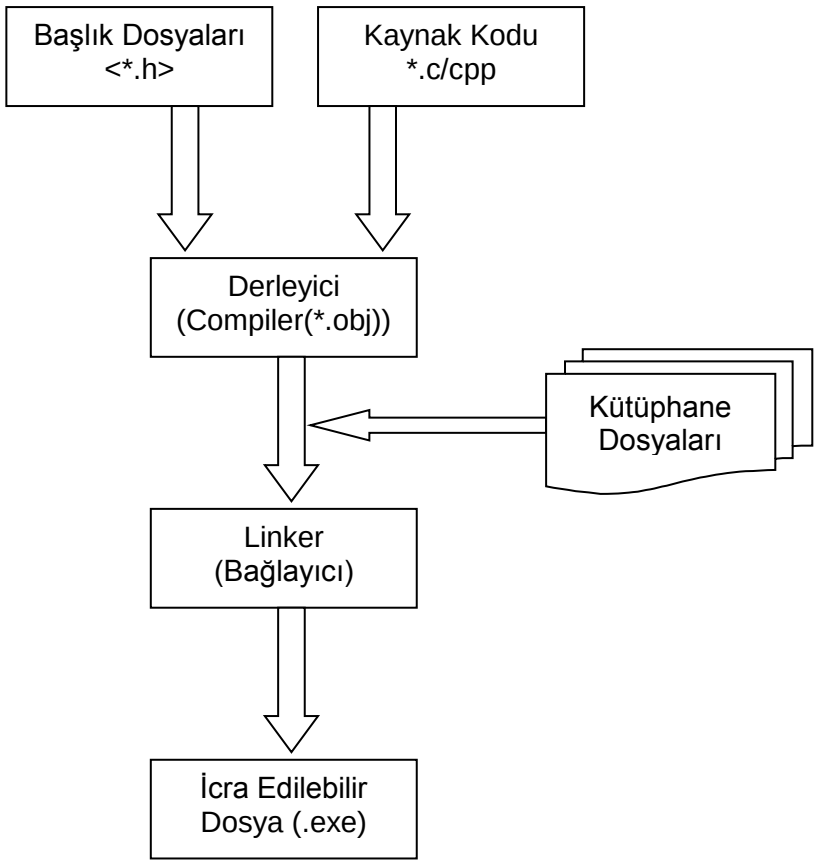
C başlangıçta bir programlama dili olarak düşünülmemişti ve özel bir amacı vardı: *UNIX işletim sistemini tasarlamak* (UNIX işletim sisteminin 1000 satırlık bölümü C ile yazılmıştır). Günümüzde Nesneye Yönelik programlama dilleri (C++, Java ve C# gibi) ve script dilleri (JavaScript, JavaApplet, PHP gibi) gibi programlama dilleri tamamen C tabanlıdır.

Kısacası Standart C dilini (ANSI C) öğrenmekle bu dillerin tamamına iyi bir hazırlık yapmış olursunuz. Bu notlarda tamamen ANSI C programlama dili konu edilmiştir.

C Programlama dilinin temel özellikleri:

- C güçlü ve esnek bir dildir. C ile işletim sistemi yazabilir, kelime işlemciler oluşturabilir veya grafik çizebilirsiniz.
- C taşınabilir bir dildir. Yani herhangi bir C kodu hiçbir değişikliğe uğramadan veya çok az bir değişimle, başka bir derleyicide derlenebilir. Örneğin, Windows işletim sistemlerinde yazılan bir C kodu, Linux, UNIX veya VAX gibi işletim sistemlerinde de derlenebilir.
- C yapısal bir dildir. C kodları *fonksiyon* olarak adlandırılan alt programlardan oluşur.

C programının çalıştırılması için gerekli adımlar:



Yukarıdaki şekilde geçen terimlerin hepsi ilerleyen bölümlerde kullanıldıkça açıklanacaktır.