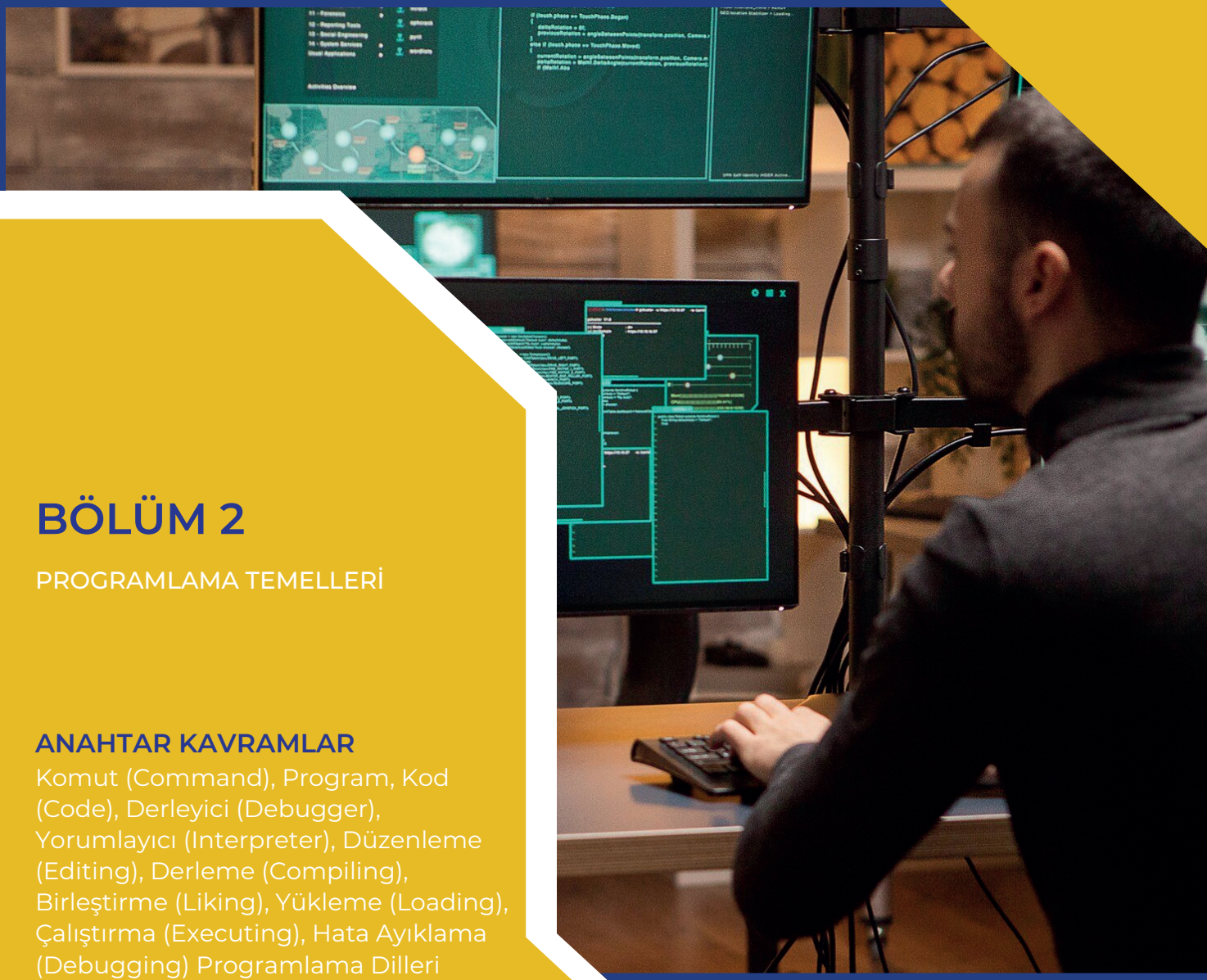


BÖLÜM 2

PROGRAMLAMA TEMELLERİ

ANAHTAR KAVRAMLAR

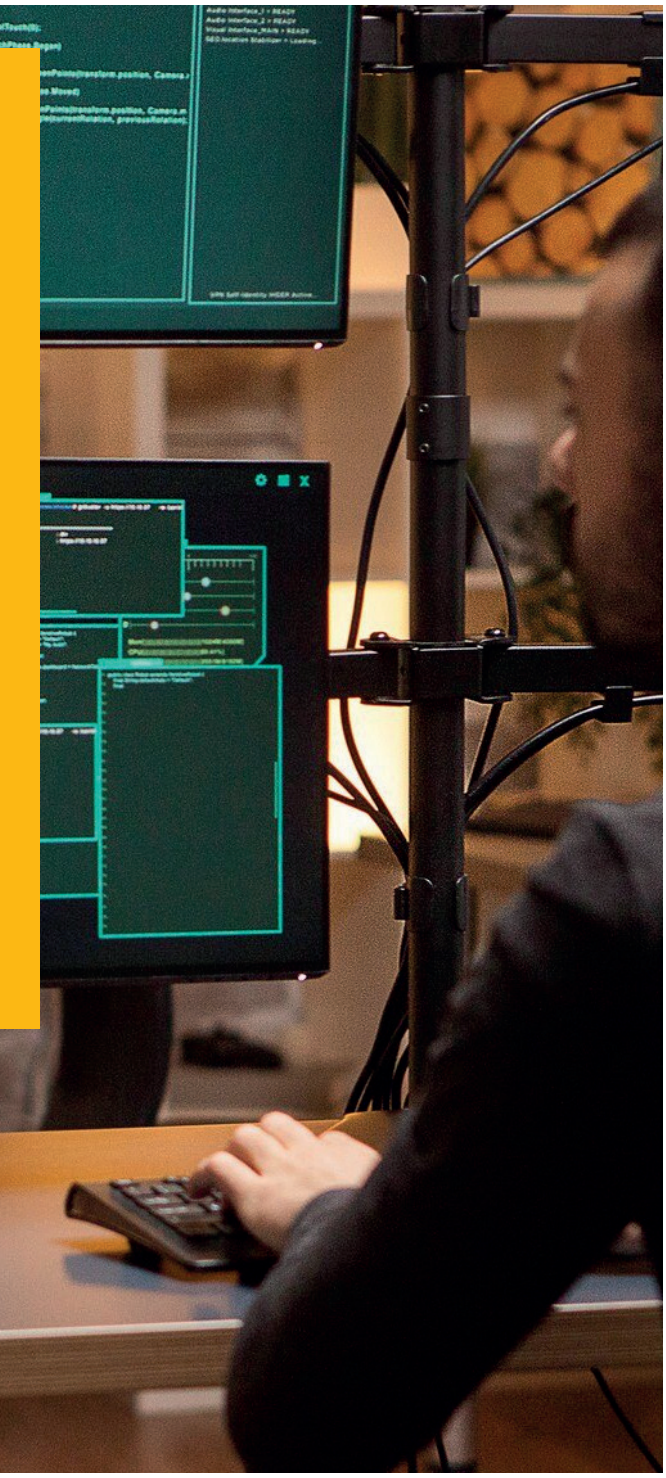
Komut (Command), Program, Kod (Code), Derleyici (Debugger), Yorumlayıcı (Interpreter), Düzenleme (Editing), Derleme (Compiling), Birleştirme (Liking), Yükleme (Loading), Çalıştırma (Executing), Hata Ayıklama (Debugging) Programlama Dilleri

ÖĞRENME HEDEFLERİ

1. Programlama hakkında gerekli teknik terminolojiyi bilme
2. Program yazma süreç ve ihtiyaçlarını bilme/ayırt etme
3. Program yazmak için kullanılan programlama dillerini tanıma

İKİNCİ BÖLÜM

GİRİŞ



Kitabımızın ikinci bölümünde programlamada geçen temel kavramlara ve programlama dillerine detaylı bir şekilde değinilmektedir. Programlama öğrenmek isteyen kişilerin diğer öğrenme yöntemlerinde olduğu gibi öncelikle terminolojiye ve program geliştireceği sisteme hem donanımsal hem de yazılımsal olarak hâkim olması gerekir. Birinci bölümünde program geliştireceğimiz bilgisayar sistemi, sayı sistemleri ve donanım üzerine bilgiler sizlere verilmişti. Bu bölüme başlamadan önce bilgisayar hakkındaki bilgilerinizi gözden geçirmeniz gerekmektedir. Bölüm 2'de ise programlamaya giriş yapılacaktır. Bu sebeple, Komut (Command), Program, Kod (Code), Derleyici (Compiler), Yorumlayıcı (Interpreter), Düzenleme (Editing), Derleme (Compiling), Birleştirme (Linking), Yükleme (Loading), Çalıştırma (Executing), Hata Ayıklama (Debugging) ve Programlama Dilleri gibi kavramlar açıklanacaktır.

DİKKAT EDELİM

Bir program yalnızca bilgisayar sistemi üzerine geliştirilmez. Günlük hayatta sıklıkla kullandığımız tabletler, telefonlar hatta su ısıtıcı üzerinde de program geliştirilebilir.

1. TEMEL KAVRAMLAR

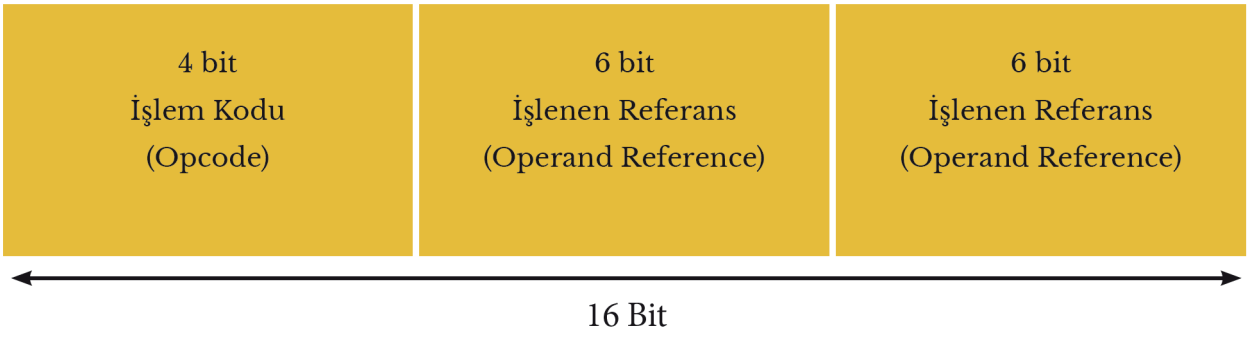
Temel kavramlar bölümü programlama ile uğraşan kişilerin gündelik hayatta kullandığı unsurları açıklamaktadır. Programlama işlemi bir dizi komutun belli bir sırayla donanımsal kaynaklar üzerinde yürütülmesi ile sağlanır. İlgili süreçte, program yazımında kullandığınız editörler (düzenleyici) ve donanımsal kaynaklar önemlidir. Bölüm 4'de C++ programlama dili kullanılarak program yazacağınız editörden bahsedilecektir. Kullandığımız işletim sisteminin sürümü ve işlemcinin türü dikkat edilmesi gereken bir hususlar. Bu sebeple, program yazan kişiler ürünlerini dağıtırken donanımsal ve yazılımsal kısıtlardan bahsederler. Ayrıca bazı programlar, kullandığımız programlama diline göre işletim sisteminden bağımsız olarak çalıştırılabilirler. Örneğin Java programlama dili ile bahsi geçen türde bir program yazılabilir. Programlama dili ile ilgili detaylı bilgilere ilerleyen kısımlarda yer verilmektedir. Öncelikle programlama için kullanılan temel terminolojik kavramlara yer verilmiştir.

EK BİLGİ

Intel, AMD, Cyrix ve Motorolla işlemcilerin mimarileri ve kullandıkları komut kümeleri aynı olmayabilir. Bu farklılıktan dolayı yazılan programların çalıştırılacak sistemlerde değişik etkilere sahip olabileceğini unutmayınız!

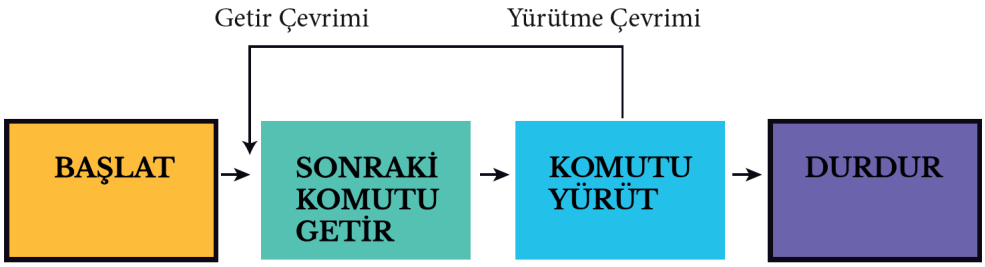
Komut (Command)

İkilik sayı sistemindeki elemanların (0 ve 1'ler) farklı şekillerde dizilişi ile komutlar oluşur. Komutlar, donanım birimlerinin tasarlanmasında ve donanımsal birimlerin birbirleri ile iletişim kurmasında kullanılır. Ayrıca her işlemci üretilirken kendisine ait özel komutlara sahip olur. Programlarda yapılması istenen işlemler işte bu komutlar aracılığıyla işlemciye iletilmektedir. İşlemci kendi üstüne gelen komutlara göre diğer donanım birimleri ile iletişime geçer. Şekil 2.1'de örnek bir komut çerçevesi gösterilmektedir.



Şekil 2.1. Basit Bir Komut Çerçevesi.

Şekil 2.1'de verilen komut çerçevesinde 16 bitlik bir komut verilmiş olup, burada ilk 4 bit komutun ne olduğunu belirlemede sonraki 6'lık bitler ise kaynak ve sonuç adresleri göstermede kullanılmaktadır. Böylece komutun ne iş yapacağı ve nerede çalışacağı işlemci tarafından anlaşılır. Şekil 2.2'de ise işlemci üzerine gelen basit bir komut çevrimi örneği verilmektedir. Çevrimin (Getir-Yürüt-Kodunu Çöz Çevrimi) animasyonlu anlatımı için bölümün sonunda verilen Faydalı Kaynaklar ve İnternet Kaynakları kısmına bakılabilir.



Şekil 2.2. Basit Komut Çevrimi.

Şekil 2.2'de komut çevrimi için kullanılan komut, işlemcide bulunan ve komut kaydı (Instruction Register IR) olarak bilinen bir kayıt defterine yüklenmektedir. Komutlarda işlemcinin gerçekleştireceği eylemi belirten bitler bulunmaktadır. Bu süreçte işlemciye iletilen komut yorumlanır ve gerekli eylem gerçekleştirilir. Genel olarak, bu eylemler dört kategoriye ayrılır (Tablo 2.1).

Tablo 2.1. Farklı Kategoride Yer Alan Komutlar.

Kategori	Açıklama
İşlemci-bellek	Veriler işlemciden belleğe veya bellekten işlemciye aktarılabilir.
İşlemci-G/Ç	Veriler, işlemci ve bir G/Ç modülü arasında aktarım yapılarak bir çevresel aygıta veya bu aygıttan işlemciye aktarılabilir.
Veri işleme	İşlemci, veriler üzerinde bazı aritmetik veya mantıksal işlemler gerçekleştirebilir.
Kontrol	Bir komut, yürütme sırasının değiştirilmesini belirtebilir.

Tablo 2.1'de kategorize edilen ve ikilik dizilerden oluşan komutlar yani makine kodu sadece donanım tarafından anlaşılabilir. Programı yazanların kolaylıkla anlayabilmesi adına bu dizeler için semboller kullanılmaktadır. Söz konusu semboller Assembly kodu olarak anılmaktadır. İki bellek konumunda saklanan numaraları toplayan bir işlem düşünelim. Matematiksel olarak, bu program x = 2, y = 5, x + y = z formüllerini temsil eder. Burada x, y ve z değişkenleri sırasıyla 13, 14 ve 15 bellek konumlarına karşılık gelsin. Programın yönergeleri Tablo 2.2'de listelenmiştir.

Tablo 2.2. Makine ve Assembly Kodu Gösterimi ile Toplama İşlemi Yapan Program.

#	Makine Kodu	Assembly Kodu	Tanım
0	001 1 000010	LOAD #2	Akümülatöre 2 değerini yükleyin
1	010 0 001101	STORE 13	Akümülatörün değerini bellek konumu 13'te saklayın
2	001 1 000101	LOAD #5	Akümülatöre 5 değerini yükleyin
3	010 0 001110	STORE 14	Akümülatörün değerini bellek konumu 14'te saklayın
4	001 0 001101	LOAD 13	Bellek konumu 13'ün değerini Akümülatöre yükleyin
5	011 0 001110	ADD 14	Akümülatöre bellek konumu 14'ün değerini ekleyin
6	010 0 001111	STORE 15	Akümülatörün değerini bellek konumu 15'te saklayın
7	111 0 000000	HALT	Yürütmeyi durdur

Tablo 2.2'de komutların makine kodu, Assembly kodu ve tanımı verilmektedir. Anlaşılacağı üzere LOAD yükleme, STORE depolama ve ADD ekleme işlemi için kullanılmaktadır. Toplama işlemini gösteren bu programda olduğu gibi komutların Assembly kodu olarak ifade edilebilmesi ile programcı verilerin bellek adresleri ve sembolleri ile yeni programlar yazabilmektedir. Bu tarz bir program yazabilmek adına işlemcilerin komut kümelerini bilmemiz gerekir. Günümüzde bu dillerin öğrenilmesi zor olmakla birlikte donanıma yakın bir dile sahip olması nedeni ile sadece donanım üzerinde program yazanlar için avantaj haline gelmektedir. Üniversitelerde bu tarz diller mikroişlemci, sistem programlama gibi isimler ile okutulmaktadır. Programlama dillerinde seviyeler kısmında bu konuya tekrar değinilecektir.

Program

Belli bir işlemi bilgisayar veya uygun yapılarda yerine getirebilmek adına kullanılan komutlar dizisine program adı verilmektedir. Bir programı oluşturabilmek adına ilgili komutların belirlenmesi ve bunların istenilen biçimde kullanılmasına da programlama adı verilmektedir. Program yazan kişiler ise Programcı olarak ifade edilir. Programın nihai hedefi son kullanıcıdır (End-User). Son kullanıcının görmediği taraftaki mimariyi oluşturanlar arka-uç programcı (Back-End), grafiksel tasarım geliştirenler ise ön-uç programcı (Front-End) olarak isimlendirilir. Bilişim dünyasında programcıların Masaüstü, Web, Mobil ve Veritabanı programlama olarak geliştirme yaptığı uygulama türleri bulunmaktadır (Tablo 2.3).

ARAŞTIRALIM

Assembly kodlarının kullanılabilmesi adına sembollerle işlem yapan programlama dilleri vardır. Emu8086 Assembly dilini Intel8086 komut kümesi ile kullandığınız bir ortamdır

Tablo 2.3. Uygulama Türleri.

Uygulama Türü	Geliştirildiği Yer	Örnek
Masaüstü	İşletim sistemi (Windows, Linux, MAC vb.)	Winrar, GIMP vb.
Web	İnternet, Intranet	https://www.ankara.edu.tr/ vb.
Mobil	iOS, Android vb.	Whatsapp, PUBG vb.
Veritabanı	Veritabanı yazılımları (Oracle, MySQL vb.)	Masaüstü, Web, Mobil uygulamaların alt yapısı için ya da Fatura, Kayıt Formu gibi raporlar

Bir programcı Tablo 2.3'de verildiği türdeki gibi uygulamalar geliştirmektedir. Burada önemli olan ihtiyacı belirlemektir. Geliştirilecek programlama son kullanıcının gereksinimlerini karşılamalıdır. İhtiyaç analizi yapıldıktan sonra başlatılan programlama sürecinde bir ya da birden fazla türde uygulama geliştirilebilir. Genelde analiz yapmak için Şelale (Waterfall), Scrum, Aşırı Programlama (Extreme Programming), Birleşik Rasyonel İşlem (Rational Unified Process) gibi Çevik (Agile) proje yönetim analizleri kullanılmaktadır.

GÜNLÜK HAYATLA İLİŞKİLENDİRELİM

Gündelik yaşantımızda ihtiyaç analizlerimizi belirlerken neler yaparız? Örneğin alış-verişe gitmeden önce; yeterince nakit ya da karşılığı olan diğer varlıklarımız (kredi kartı, çek, senet vb.) var mı?, nelere ihtiyaç var?, alış-verişte öncelik hangilerine verilmeli? şeklinde olabilir. Programlamaya başlamadan önce sizce nasıl analizler yapılır? Program yazma için gerekli olan analiz yöntemlerini inceleyelim.

Kaynak Kod (Source Code)

Kaynak kod bir programlama diliyle yazılan programların işlemcinin anlayabileceği makine diline dönüştürülerek işlenip yorumlanmasından önce programcının okuyabildiği ve üzerinde çalışabildiği kodlar olarak tanımlanır. Buradan da anlaşılacağı üzere kaynak kod programcı dostu bir yazıma sahiptir. Bu kodlar evrensel bir dilde tanımlanmalıdır. Böylece dünyanın herhangi bir yerindeki diğer programcılar tarafından incelenmesi mümkün olacaktır. Bu bağlamda, yeni bir özellik ekleme, sorun saptama ve farklı programlar geliştirmeye yardımcı olunmaktadır. Evrensel dilde anlaşmak programlama dillerinin doğmasına yol açmaktadır. Dersimizde kullanacağımız C++ böyle bir dil olup, işletim sistemlerinin yazılmasından, çeşitli amaçlara hizmet eden pek çok programa kadar geniş bir yelpazeye sahiptir.

Her kaynak kod bir dosyada muhafaza edilir. Örneğin;

- C++ için <dosya_adı>.cpp
- C# için <dosya_adı>.cs
- Java için <dosya_adı>.java
- Python için <dosya_adı>.py

Kaynak kodlar bir bilgisayar üzerinde direkt çalıştırılamazlar. Bunun için gerekli bir takım yazılımların yüklenmesi gerekmektedir. Bölüm 4'de detaylı bir şekilde anlatılacaktır.

Amaç Kod (Object Code)

Amaç kod ise işlemcinin komut kümeleri kullanılarak kaynak kodun makine dile çevrilmesidir. Böylece programcı hem kendi anlayabildiği bir dilde program yazmış olacak hem de derlenen program makinenin anladığı bir dile dönüştürülecektir. Bu tarz dosyalar genelde .obj uzantılı dosyalarda ya da obj isimli klasörlerde tutulmaktadır. Ancak düzenlenmesi için oluşturulan dosyalar değildir. Amaç kod yazılan kaynak koddan birleştirme işlemi ile çalıştırılabilir bir dosya oluşturmak adına kullanılır.

Derleyiciler (Compiler)

Derleyiciler programlama ile uğraşan kişilerin en önemli aracıdır. Söz konusu süreçte kullanılan programlama dili hangi derleyicileri kullanacağımızı da belirlemektedir. Herhangi bir programlama diliyle yazılmış olan kaynak kodumuzun, makine diline yani amaç programa dönüştüren özel programlara derleyici adı verilmektedir. Burada yazılan kaynak kodun tamamı tek seferde derlenmektedir. Bu anlamda, hem çevrimiçi hem de çevrim dışı kullanabilen pek çok derleyici bulunmaktadır. Java, C, C++, COBOL, Fortran çevrim dışı yani işletim sistemine yüklenerek kullanabilecek programlama dilleridir. Yanı sıra https://www.onlinegdb.com/online_c++_compiler web adresinde olduğu gibi C++ ile yazılmış kodları derleyen çevrim içi editörler de bulunmaktadır. Çevrim içi editörlerde bilgisayarımıza bir yazılım yüklemeye gerek kalmadan da programlarımızı derleyebiliriz. Bu tarz

uygulamalarda sadece aktif internet bağlantısına ihtiyacımız bulunmaktadır.

Yorumlayıcılar (Interpreter)

Evrensel dil kullanılarak oluşturduğumuz kaynak kodun ilk satırdan başlayarak satır satır makine diline çevrilmesine ve yazılan kodun çalıştırılmasında etken olan araca yorumlayıcı adı verilmektedir. Genelde web uygulaması için yazılan programlarda yorumlayıcılar kullanılmaktadır. Yazılan kaynak kodun herhangi bir satırında oluşan hata, programın bütününün çalışmasında sorun yarattığı için yorumlayıcılar bazı küçük hataları görmemektedir. Çünkü hata olan satırdan sonraki yazılan kaynak kodlar çalıştırılmayacaktır. Bu anlamda yorumlayıcı kullanılarak yazılan programlarda çok dikkatli olunması gerekmektedir. Bilgisayarlarımızda kullanıldığımız tarayıcılar aynı zamanda birer yorumlayıcıdır. Örneğin;

- Mozilla Firefox
- Google Chrome
- Microsoft Edge/Microsoft Internet Explorer
- Opera
- Netscape

Bunun dışında Perl, PHP, Python, Ruby ve JavaScript, kullanılan diğer yorumlayıcılar arasındadır.

Derleyici ve Yorumlayıcı Arasındaki Farklar

Programcılar sadece kaynak kod için kullanılan kısımları anlayabilmekte ya da düzenlemektedir. Bilgisayar veya diğer donanımsal yapılar ise yalnızca ikilik dilde yazılmış kodları anlayabilir. Bu nedenle program geliştirilen yapılar için derleyici ya da yorumlayıcıya ihtiyaç bulunmaktadır. Programcıların kullandığı sistemler ya yorumlama ya da derleme üzerine çalışmaktadır. Kullanılan sistem kaynak kodu makinenin anlayabileceği koda yani makine diline çevirmektedir.

- Yorumlayıcı programcının yazdığı her bir satırdaki kod ifadesini okur, doğrudan dönüştürür ve yürütür.
- Derleyici programcının yazdığı bütün satıları tek seferde işletim sistemi tarafından çalıştırılabilen yerel (derlenmiş) koda çevirir. Bunun için derleme işlemi ile birlikte yürütülebilir (.exe uzantılı) bir dosya meydana getirir.
- Derleyici yorumlayıcıya göre kaynak kodu satır satır işletmediği için daha hızlı çalışmaktadır. Çünkü yorumlayıcı derleyicinin aksine tüm programı taramak yerine her seferinde bir ifadeyi çevirmektedir.
- Derleyiciler belleği daha verimli kullanma eğiliminde olan yorumlayıcıların aksine bağlanmak için daha fazla bellek gerektiren ara nesne kodu üretmek zorundadır.
- Bir yorumlayıcı, kodu tek bir işlemde okuyup çalıştırdığından, komut dosyası oluşturmada ve diğer küçük programlar için çok kullanışlıdır.
- Yorumlayıcı yazılan kaynak kodu satır satır çalıştırdığı için hatalı olan kısımlardan sonrasını çalıştırmaz. Bundan dolayı programın diğer kısımlarında yazılmış doğru kodlar çalıştırılmayacaktır. Derleyici ise yazılan kaynak kodun tamamını tek seferde işlettiği için hatalı kısımlar doğru yazılan kaynak kodları etkilemeyecektir.

Düzenleme (Editing)

Bulunduğumuz çağda kod yazmak oldukça önemli bir hale gelmiştir. Bugün pek çok disiplinde işleri kolaylaştırmak adına farklı görevleri yerine getiren masaüstü, mobil ya da web uygulamaları geliştirilmektedir. Günümüzde nesnelerin interneti (IoT) kavramı ile de program geliştirilen araçların sayısı artmıştır. Programcı programını yazmadan önce bir kod düzenleyi-

ci belirlemelidir. Tümleşik geliştirme ortamları (Integrated Development Environment IDE) kodlama, derleme ve hataları tespit edebilme adına pek çok süreçte kolaylık sağlamaktadır. Hatta artık pek çok IDE ile birden fazla programlama diline destek verilmektedir.

Microsoft firmasının geliştirdiği Visual Studio, 36 farklı programlama dilini destekler. Yerleşik diller arasında C, C++ , C++/CLI, Visual Basic .NET, C# , F# , JavaScript , TypeScript , XML , XSLT , HTML ve CSS bulunur. Python, Ruby, Node.js gibi diğer diller için de destek sunulmaktadır. Bazı diller ise eklentiler aracılığıyla kullanılabilmektedir. Visual Studio'nun en temel sürümü olan Topluluk (Community) sürümü ücretsizdir.

Popüler olan düzenleyiciler,

- Visual Studio Code
- Cygwin
- Notepad++
- Sublime Text
- UltraEdit
- Atom
- Brackets
- GNU Emacs
- BBEdit
- CodePen

Derleme (Compiling)

Derleme işleminde, kullanılan programlama dilinin (C++, C# vb.) ön işlemcisi tarafından oluşturulan kaynak kod derleyici tarafından programın yazıldığı makinedeki işlemcinin komut kümesine uygun makine diline ve ardından işletim sistemine uygun amaç koda dönüştürülür. Aynı süreçte birleştirme işlemi ile çalıştırılabilir bir dosya oluşur.

Birleştirme (Linking)

Amaç kodun, son kullanıcının kullanabileceği yani yürütülebilir bir program oluşturması adına gerekli destekleyici kodla birleştirilmesi gerekmektedir. Bu adımda genellikle gerekli olan kitaplıklara ekleme işlemi bulunmaktadır. Birleştirme işlemi derleme işlemi ile birlikte tetiklenerek, amaç kod ile fonksiyonların ait oldukları kütüphaneler birleştirmede kullanılmaktadır. İşlem sonunda çalıştırılabilecek bir program dosyası oluşur.

Yükleme (Loading)

Programcı tarafından yazılan kodun yürütülmesinde çalışılan sistemin birincil belleği kullanılmaktadır. Bu sebeple programın öncelikle buraya yüklenmesi gerekmektedir. İşletim sistemleri yükleyici adı verilen bir bileşene sahiptir ve çalıştırılmaya ihtiyaç duyulan program ve gerekli kütüphane dosyası depolama biriminden okunarak bilgisayarın birincil belleğine yüklenir.

Çalıştırma (Executing)

Çalıştırma işleminde, derlenen/yorumlanan kaynak koddan üretilen makine dili programı ya da makine dili komutu işlemci aracılığıyla bellekten alınarak yürütme işlemi yapılmaktadır.

Hata Ayıklama (Debugging)

Derlenen/yorumlanan kod çalıştırılmadan önce varsa hatalarından ayıklanması gerekmektedir. Kullandığımız IDE aracılığıyla bu tarz bir işlem yapabiliriz. Burada üç türde hata ile karşılaşırız.

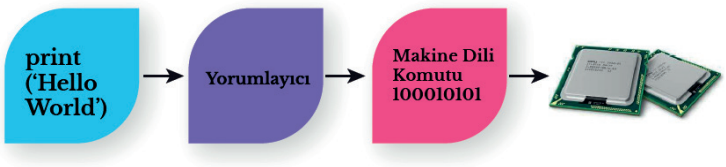
- Sözdizimi hatası: Dilbilgisi hatası gibidir. Programcının kullandığı programlama diline göre belli yazım kuralları vardır. Bu kurallara göre bir noktalama işaretinin unutulması, kelimelerin yanlış ya da eksik yazılması sözdizimi hatasıdır. Söz konusu hatalar, program derlen-

meden önce bilgisayar tarafından otomatik olarak algılanabilir.

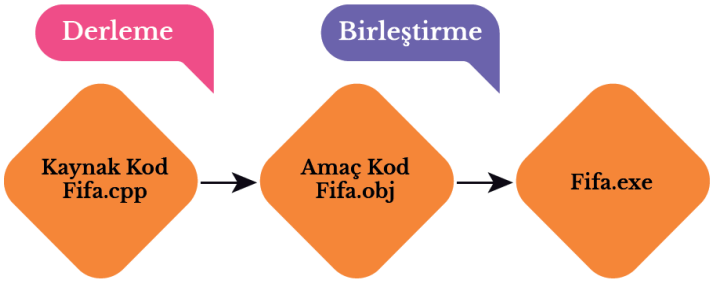
• Mantık hatası: Bu hata türü sadece programcının tespit edebileceği türdendir. Yani temelde bilgisayarın çökmesine veya bir hata mesajı görüntülemesine neden olmayan, ancak program çalıştığında istenen sonucu göremediğimizde oluşan hatalardır. Örneğin, matematiksel bir işlemde bir değeri 10 ile çarpmak yerine, yanlışlıkla 100 ile çarpmamız. Bilgisayar sonuçta işlemi yapacak ve hata vermeyecektir. Fakat sonuç yanlış hesaplanmış olacaktır.

• Çalışma zamanı hatası: Program çalışırken bazı geçersiz komutlar yürütüldüğünde, programın çökmesi ve/veya bir hata ekranı açılması durumudur. Örneğin, haftanın her günü için 7 elemanlı bir verinin program ile belleğe aktarıldığını düşünelim. Program bu dizinin olmayan 8. konumundaki içeriğe erişmeye çalışırsa, komut geçersiz olduğundan ve bilgisayar talimatı nasıl işleyeceğini bilmediğinden bir çalışma zamanı hatası üretebilir. Bilgisayarların olası çalışma zamanı hatalarını önceden algılaması temelde imkânsızdır. Bunlar, programı test ederken tesadüfen keşfettiğiniz hata türleridir.

Şekil 2.3'de yorumlayıcı kullanan, Şekil 2.4'de ise derleyici kullanan programlama durumlarına diyagram kullanılarak örnek verilmiştir.



Şekil 2.4. Derleyici ile Programlama Süreci



Şekil 2.4. Derleyici ile Programlama Süreci.

2. PROGRAMLAMA DİLLERİ

Programlama dili sisteme girdi olarak verilen öğeleri çeşitli makine kodu çıktılarına dönüştüren ve kendine has kurallar dizisine sahip yazılımlardır. Programlama dilleri, bilgisayarın avantajlarını kullanarak arzu edilen işi hızlı ve verimli yerine getirmekte kullanılmaktadır. Ancak her dilin farklı kullanım amaçları olduğundan yapılmak istenen işe göre tercih edilmesi gerekir. Bu anlamda donanım programlamak için makine diline yakın diller ya da otomasyon geliştirmek adına kullanıcı dostu diller bulunmaktadır.

Programlama süreci 1800'li yıllara dayanmaktadır. Geçmişte çeşitli müzik enstrümanları ve dokuma tezgâhlarını programla fikri ile bir dizi komut kümesi kullanılmıştır. O zamandan süre gelen gelişimler programlama dillerini hemen hemen her türlü araç-gereçte kullanmaya imkân sağlamaktadır.

Günümüzde popüler olarak kullanılan programlama dilleri:

• **Python:** Programlamaya yeni başlayanlar için kullanılan ve sözdizimi basit bir dildir. Geniş

bir kitaplığı bulunmaktadır. C ve C++ gibi programlama dilleriyle kolay entegre olmaktadır. Yapay zekâ, finansal hizmetler ve veri bilimi dâhil olmak üzere çok çeşitli uygulamalarda kullanılmaktadır. YouTube, Google ve iRobot makineleri dâhil olmak üzere Instagram ve Pinterest gibi sosyal medya siteleri Python üzerine kuruludur.

• **JavaScript:** Web ve mobil uygulama oluşturmak için tercih edilen bir programlama dilidir. Son kullanıcının bir uygulama indirmesini gerektirmeyen tarayıcı tabanlı uygulamaların geliştirilmesine olanak tanır. Son zamanlarda çeşitli eklentiler ile JavaScript'in işlevselliği artırılmıştır. Bu anlamda, Node.js, Vue.js, AngularJS, React gibi çeşitli çerçeveler (frameworks) oluşturulmuştur. Bu çerçeveler ile programcının uygulamayı bir cihaza duyarlı olarak kodlaması mümkündür. Bu yanıt verebilirlik, üst düzey bir makine dili kullanma konusunda oldukça popüler olmasının bir başka nedenidir.

• **C:** Genelde programlamayı yeni başlayanların tercih ettiği ve ilk geliştirilen programlama dillerinden birisidir. 1972 yılında geliştirilmiştir. Bugün pek çok üniversite programlamaya giriş dersinde C ya da C++ programlama dilini tercih etmektedir. Python, Ruby ve PHP gibi son zamanlarda popüler olan programlama dillerine temel teşkil etmektedir. Her tür cihazda çalışabildiğinden, genellikle otomobillerdeki gömülü cihazlar ve sağlık hizmetlerinde kullanılan tıbbi cihazlar gibi donanımları programlamak için kullanılır.

• **C++:** Genelde işletim sistemleri programlamak için tercih edilmektedir. Zengin bir kütüphanesi bulunmaktadır. Matematiksel simülasyonlarda ve bilgisayar oyunlarında da kullanılması sık rastlanan durumdur. Çoklu cihazlarda kullanımı ve çeşitli platformları desteklemesi üstünlükleri arasında sayılır.

• **Java:** 1991 yılında geliştirilmiş olup, bugün halen dünya çapında büyük işletmeler tarafından tercih edilmektedir. İstemci-sunucu mimarisine uygun bir programlama yapısı vardır. Bu dilde yazılan uygulamalar Java'yı destekleyen herhangi bir platformda çalıştırılabilir. İş uygulamalarıyla birlikte Java, Android mobil işletim sisteminde de yaygın olarak kullanılmaktadır.

• **C#:** Microsoft tarafından 2002 yılında C'nin alt yapısı kullanılarak geliştirilmiştir. Windows, Android, Azure, iOS, Linux, macOS, tvOS ve Xbox platformlarına destek vermektedir. Bulut, Konsol, Masaüstü, Tarayıcı Uzantısı, Oyun, Nesnelerin İnterneti, Makine Öğrenmesi, Mobil vb. projelerde tercih edilmektedir. Microsoft .NET IDE'si ile tamamen entegredir. Büyük bir kod kitaplığı olup üzerinde çeşitli veri türleri barındırmaktadır. Söz konusu özelliği ile sistem kaynaklarını daha etkin kullanır. Son zamanlarda Mono adlı .NET Framework uzantısını kullanan mobil cihazlar ve video oyun konsolları için yazılan çeşitli C# uygulamaları göze çarpmaktadır.

• **GO:** Microsoft'tan sonra piyasaya hâkimliği ile tanınan Google firması tarafından geliştirilen bir dildir. Özellikle ağdaki cihazların haberleştirildiği dağıtık sistemlerde tercih edilmektedir. Söz dizimi basit olup, bellek güvenliği ve yönetimi konusunda etkindir. Bugün C/C++ gibi dillerle yarışır durumdadır. 2007 yılında geliştirilmiş olup açık kaynak bir programlama dilidir.

• **R:** İstatistikçilerin kullandığı bir numaralı programlama dili olma yolunda ilerlemektedir. Büyük küme veri analizleri, makine öğrenimi gibi çağımız uygulamalarında da sıklıkla tercih edilmektedir.

• **Swift:** Apple firmasının 2014 tarihinde başlattığı bir proje olup, iOS ve macOS uygulamaları geliştirmek için tercih edilmektedir. Nesneye yönelik bir programlama dilidir. Swift ile Apple'ır tüm ürünlerinde çalıştırılabilir hızlı ve etkin uygulamalar geliştirilebilmektedir. Swift'in dezavantajı, iOS 7 sonrası sürümleri desteklemesidir.

• **PHP:** 1995 yılında geliştirilmiştir. Facebook, WordPress ve Wikipedia gibi içerik odaklı web sitelerinin yazılımında kullanılan, dinamik bir programlama dilidir. Hata ayıklaması kolaydır.

Ancak son zamanlarda Python ve JavaScript gibi diller popülaritesini azaltmıştır.

Yukarıda bahsedilen programlama dillerinin yanı sıra Rust, Ruby, Perl ve MATLAB gibi daha eç çok dil programcıların ilgisini çekmektedir. Burada önemli olan programlama yapmak istediğiniz göreve en uygun dili belirlemektir. Programcılar ayrıca dillerin seviyelerine göre de tercih de bulunmaktadır. İlerleyen kısımda programlama dili seviyelerine de değinilecektir.

3. PROGRAMLAMA DİLLERİNİN HEDEFLERİ

Programlama dillerinin belirgin amacı, temelde bilgisayara talimat sağlamaktır. Bu nedenle, programlama dilleri, daha yüksek derecede kesinlik ve tamlık gerektirmeleri bakımından diğer ifade biçimlerinden farklıdır. Bu bağlamda programcının programlamaya başlamadan önce hedeflerini tam olarak belirlemesi gerekmektedir.

Basitlik ve Okunabilirlik

Yazılan programlar ortalama bir programcı tarafından anlaşılabilir ve okunması kolay olmalıdır. Dolayısıyla ideal bir programlama basit ve okunabilir kriterlere göre tasarlanmalıdır. Program geliştiren firmalar bu bağlamda bir standart belirlemektedirler. Değişken ve fonksiyon isimlerinin nasıl tanımlanacağının belirlenmesi, açıklayıcı yorum satırlarının eklenmesi, yazılan her modül için doküman hazırlanması örnek olarak verilebilir. Günümüzde özellikle açık kaynak kodlamanın hâkim olması, yazılımın bu hedefi ile de örtüşmektedir.

Güvenirlik ve Destek

Yazılan programların bir test etme süreci bulunmaktadır. Test aşamasında yazılımın farklı platformlara uygulanıp uygulanamayacağı, aynı girdi verileri ile çalıştırıldığında aynı sonuca ulaşıp ulaşmadığı yazılımın destek mekanizması ve güvenirliği açısından önemlidir.

Sürdürülebilirlik

Yazılım geliştirildikten sonra da programlama işlemi bitmemektedir. Bu bağlamda, programların değişen koşullara uygun parametrik yapısının olması yazılımın sürdürülebilirliğin gereğidir. Gelişen bilişim dünyasında programların üzerinde çalıştırıldığı işletim sistemleri de zorunlu olarak değişmektedir. Dolayısıyla program yazılırken kaynak kodun o zamanki sisteme göre yeniden yapılandırılması gerekmektedir. Ayrıca yazılan programlar için verilen eğitimler sürdürülebilirlik noktasında diğer önemli bir hususdur.

Verimlilik

Programların sistem gereksinimleri ve çalışma hızları verimliliğine etki eden unsurlardır. Günümüzde pek çok işlemin çözümleri yazılan programlarla sağlanmaktadır. Bu durum programlara düşen rolü artırmaktadır. Dolayısıyla yüksek verimliliğe sahip yazılımlara ihtiyaç duyulmaktadır. Yüksek verimliliğin sağlanabilmesi adına da sistemin kaynakları optimal olarak kullanılmalıdır. Bellek gereksinimi, güvenlik ve yönetim programcının dikkat etmesi gereken başlıca konulardır.

4. PROGRAMLAMA DİLLERİNDE SEVİYELER

Programlama dilleri geliştirildiği platformun diline yakınlığına göre seviyelere ayrılmaktadır. Kullanılan dil ne kadar kullanıcı dostu ise dilin seviyesi de artmaktadır. Ancak burada önemli olan yazılan programın amaca uygunluğudur. Düşük seviyede diller her ne kadar kullanıcı dostu olmasa da donanımın programlanmasında önemlidirler.

Makine Dilleri

Makine dilleri, programı yazdığımız bilgisayarın işlemcisi tarafından doğrudan yürütülebilen dillerdir. Temelde bir bit deseni olarak ifade edilmektedir. Bu desenler sekizli ya da onaltılı sayı sistemleri ile formüle edilir. Oluşturulan her bit deseni işlemciye ait temel işlemlerden birisine karşılık gelmektedir. Dolayısıyla makine dilleri bir işlemci ailesine özgüdür. Bu bağlamda, bir işlemci ailesine ait olan bir bit deseni diğer işlemci ailelerinde kullanılamaz ya da uyumluluk modu içeren ek donanıma sahip işlemciler tarafından kullanabilecektir. Makine dilleri işlemciyi üreten firma tarafından tanımlanmaktadır. Yaygın olarak kullanılan makine kodları komut setleri Tablo 2.4'de verilmiştir. Şekil 2.5'de ise Fibonacci sayılarını hesaplamak için yazılmış örnek bir makine dili programı bulunmaktadır.

EK BİLGİ

Fibonacci sayıları 1, 1, 2, 3, 5, 8, 13 ve 21 şeklinde giden ve altın oran olarak bilinen bir dizidir. Burada her sayı kendisinden önce gelen sayı ile toplanarak büyümektedir. 1202 yılında, İtalyan matematikçi Leonardo Fibonacci tarafından bir probleme çözüm bulmak için geliştirilmiştir.

Tablo 2.4. Makine Kodları Komut Kümeleri.

Komut Kümesi Ailesi	Versiyonlar
ARM	16, 32, 64 bit
DEC	PDP, LINC, DECSYSTEM, VAX, Alpha
Intel 8008, 8080 ve 8085	Zilog Z80
x86	Intel 8086, 8088, 80186, 80286, 80386, AMD
IBM	
MIPS	
Motorola 6800	
MOS Technology 65xx	6502, 6510, 65816/65802
National Semiconductor NS320xx	
POWER	PowerPC, Power ISA
Oracle SPARC	
UNIVAC	490, 492, 494, 1230, 1101, 1103, 1105, 1100/2200 series

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD989
C14AEBF1 5BC3
```

Şekil 2.5. Fibonacci Sayısını Hesaplayan 32-Bit x86 Makine Kodunun Onaltılık Gösterimi.

Düşük Seviyeli/Çevirici Diller

Makine dillerini çok yakın olan bir dil grubudur. İşlemcinin komut kümesine yapısal olarak benzeyen semboller ile kullanılmaktadır. Bu sebeple, düşük seviyeli dillerin kullanımında derleyici/yorumlayıcıya ihtiyaç olmaksızın donanımın ihtiyacı olan makine koduna dönüştürme işlemi yapılabilir. Çevirici (Assembler) sayesinde elde edilen kod, işlemci üzerinde doğrudan çalışmaktadır. Düşük seviyeli dillerde bellek kullanımı düşük olduğu için hızlıdır. Kullanıcı dostu olmadıklarından öğrenilmesi zordur. Ancak donanım programlamada önemli bir yere sahiptir. Şekil 2.6'da çevirici dil ile yazılmış örnek bir program bulunmaktadır.

```
_fib: movl
        $  1 ,  %eax
        xorl  %ebx ,  %ebx
.fib_loop:
        cmpl  $1 ,  %edi
        jbe  .fib_done movl
        %  eax ,  %ecx
        addl  %ebx ,  %eax
        movl  %ecx ,  %ebx
        subl  $1 ,  %edi
        jmp  .fib_
.fib_done:
        ret
```

Şekil 2.6. Fibonacci Sayısını Hesaplayan AT&T Sözdizimi.

Yüksek Seviyeli Diller

Makine dili ve çevirici dil dışında kalan ve kullanıcı dostu olarak ifade edilen dillerdir. Söz konusu programlama dilleri yüksek seviyeli olarak geçmektedir. Ancak bu programlama dilleri her ne kadar kullanıcıya yakın olsa da işin işlemci tarafından yürütülmesinden dolayı yazılan programların donanımın anlayacağı dile dönüştürülmesi gerekmektedir. Bu bağlamda devreye derleyici ve yorumlayıcılar girmektedir. Şekil 2.7'de yüksek seviyeli dile örnek bir program verilmiştir.

İlk yaygın olarak kullanılan yüksek seviyeli dil Fortran'dır. Yüksek seviyeli dillerde diğer dil ailesinden farklı olarak bellek adresleri ve çağrılar ile uğraşmak yerine değişken, dizi, nesne, aritmetik ve mantıksal ifadeler, fonksiyonlar ve döngüler gibi işlemlere geçilmiştir. Bölüm 3'de sözde kod yazılarak bu işlemlere giriş yapılacaktır. Kalan diğer bölümlerde ise C++ söz dizimi kullanılarak programlama öğretilecektir.

```
unsigned int fib(unsigned int n) {
   if (!n)
       return 0;
   else if (n <= 2)
       return 1;
   else {
       unsigned int a, c;
       for (a = c = 1; ; --n) {
           c += a;
           if (n <= 2) return c;
           a = c - a;
       }
   }
}
```

Şekil 2.7. Fibonacci Sayısını Hesaplayan C Programı.

BÖLÜM ÖZETİ

Bu bölümde programlamayı öğrenmek isteyen bireylere temel kavramları kazandırmak ve farklı programlama dillerinin sunduğu olanakları göstermeyi amaçlamaktadır. Genel olarak, geliştirilmek istenen uygulamanın türü ve kullanılacak programlama dili, ihtiyaç analizi sürecinden sonra belirlenmektedir. Bu süreç tamamlandıktan sonra programlama dilinin seviyesine ve belirlenen seviye için hangi dilin tercih edilmesi gerektiğine karar verilir. Bu karar verme sürecinde verimlilik, sürdürülebilirlik ve sağlanabilecek destek gibi faktörler önemli rol oynamaktadır.

Programlama alanında masaüstü, web, mobil ve veritabanı uygulamaları geliştirme konuları sıklıkla ele alınmaktadır. Web uygulamaları geliştirirken Python, JavaScript ve PHP; masaüstü yazılımlarında C# veya Java; veri analizi ve bilimsel hesaplamalarda ise R ve MATLAB öne çıkan diller arasında yer almaktadır. Gömülü sistem uygulamalarında C, C++ ve Rust sıklıkla tercih edilmektedir. Bulut tabanlı uygulamalarda Go ve Scala gibi diller dikkat çekerken, mobil cihazlar için uygulama geliştirme süreçlerinde Swift ve Kotlin gibi diller yaygın olarak kullanılmaktadır.

Programlama dili seçiminde, uygulama türü ve ihtiyaç duyulan performans kriterleri belirleyicidir. Farklı dillerin sunduğu özellikler ve kullanım alanları göz önünde bulundurularak, hedeflere en uygun dil veya diller tercih edilmelidir.

GÖZDEN GEÇİRELİM

1. **Aşağıda verilenlerden hangisi ikilik sayı sistemindeki elemanların farklı diziliş-leri ile oluşturulmaktadır?**
   a. Komut
   b. İşlemci
   c. Bellek
   d. G/Ç Modülü
   e. Scrum

2. **Derleyici ve yorumlayıcı için verilen bilgilerden hangisi yanlıştır?**
   a. Yorumlayıcı satır satır işler.
   b. Derleyici tüm kaynak kodu tek seferde işler.
   c. Derleyici, belleği yorumlayıcıdan daha verimli kullanır.
   d. Yorumlayıcıda hatalı satırdan sonrası işletilmez.
   e. Derleyici hatalı satırlar olsa da diğer satırları çalıştırır.

3. **Microsoft Visual Studio'da yerleşik olarak hangi programlama dili bulunmaz?**
   a. C
   b. Visual Basic
   c. F#
   d. C#
   e. Linux

4. **Aşağıdakilerden hangisi programlama dili için kullanılan bir düzenleyici (editör) değildir?**
   a. Cygwin
   b. Linux
   c. UltraEdit
   d. Atom
   e. Notepad++

5. **Akümülatöre değer yükleyen Assembly kodu aşağıdakilerden hangisidir?**
   a. Load
   b. Store
   c. Add
   d. Halt
   e. Mov

6. **Bir programlama diliyle yazılan programların işlemcinin anlayabileceği makine diline dönüştürülerek işlenip yorumlanmasından önce programcının okuyabildiği ve üzerinde çalışabildiği kısımlar aşağıdakilerden hangisidir?**
   a. Yorumlayıcı
   b. Derleyici
   c. Amaç Kod
   d. Kaynak Kod
   e. Birleştirme

7. **İstatistik için kullanılan programlama dili hangisidir?**
   a. Go
   b. R
   c. Scala
   d. Kotlin
   e. C++

8. **Aşağıda verilenlerden hangisi makine kodları için kullanılan komut kümesi ailelerinden değildir?**
   a. Motorola 6800
   b. ARM
   c. Java
   d. IBM
   e. Intel 8008

9. **İlk yaygın olarak kullanılan yüksek seviyeli dil hangisidir?**
   a. JavaScript
   b. PHP
   c. Java
   d. C#
   e. Fortran

10. **Aşağıda verilen programlama dili hedeflerinden hangisi sürdürülebilirlik ilkesine uygundur?**
   a. Programların değişen koşullara uygun parametrik yapısının olması
   b. Programın farklı platformlara uygun olması
   c. Programın basit ve anlaşılır olması
   d. Programın sistem gereksinimi ve çalışma hızı
   e. Hiçbiri

Yanıt Anahtarı: 1-A, 2-C, 3-E, 4-B, 5-A, 6-D, 7-B, 8-C, 9-E, 10-A

Kaynakça

William, S. (2016). Computer organization and architecture: designing for performance, Pearson.

Demir, S., Eryiğit, R., Ar, Y., Tuğrul, B., Gök, O., Kul, S., . . .

Yağanoğlu, M. (2019). Programlama Temelleri. Erzurum: Atatürk Üniversitesi Açıköğretim Fakültesi.

Kaleli, C., Bilge, A., Ergin, S., & Şora Günal, E. (2016). Bilgisayar ve Programlamaya Giriş. Eskişehir: Anadolu Üniversitesi.

En.wikipedia.org. 2022. Low-level programming language - Wikipedia. [online] <https://en.wikipedia.org/wiki/Low-level_programming_language> [Erişim Tarihi 30 Ocak 2022].

Faydalı İnternet Kaynakları

Getir-Yürüt-Kodunu Çöz Çevrimi (https://www.youtube.com/watch?v=04UGopESS6A)

Assembly Dili (https://tr.wikipedia.org/wiki/Assembly)