

A thick dark blue vertical bar is positioned on the left side of the page. A blue arrow-shaped banner points to the right from this bar, containing the date. Below the banner, several thin, curved lines in dark blue and light grey sweep upwards from the bottom left corner.

29 Ekim 2017

# Bilgisayar Bilimi

Kur1 Ders Notları

Algoritmalar

Ahmet Tuna POTUR

## İÇİNDEKİLER

<b>İçindekiler .....</b>	<b>i</b>
<b>I. Bölüm: Algoritmalar .....</b>	<b>1</b>
<b>1. Etik, Güvenlik ve Toplum .....</b>	<b>1</b>
1.1. Etik Değer .....	1
1.2. Bilgi Güvenliği, Sayısal Dünyada Parola Yönetimi .....	1
<b>2. Problem Çözme .....</b>	<b>2</b>
2.1. Tilki, Kaz ve Mısır Çuvalı .....	2
2.2. Problem Çözme Teknikleri .....	3
2.2.1. Her Zaman Bir Planınız Olsun.....	3
2.2.2. Problemi Tekrar İfade Edin .....	3
2.2.3. Problemi Küçük Parçalara Ayırın.....	3
2.2.4. Önce Bildiklerinizden Yola Çıkın .....	3
2.2.5. Problemi Basitleştirin .....	3
2.2.6. Benzerlikleri Arayın .....	3
2.2.7. Deneme Yapın.....	3
2.2.8. Asla Vazgeçmeyin .....	3
2.3. Yazılım Geliştirme Süreci .....	4
2.3.1. Analiz Aşaması .....	4
2.3.2. Tasarım Aşaması .....	4
2.3.3. Kodlama(Geliştirme) Aşaması.....	4
2.3.4. Test Aşaması .....	4
2.3.5. Çalıştırma (Devreye Alma) Aşaması .....	4
<b>3. Bilgisayarın Çalışma Mantığı.....</b>	<b>4</b>
3.1. Bilgisayar Nedir? .....	4
3.1.1. Giriş Birimi .....	5
3.1.2. Bellek Birimi.....	5
3.1.3. İşlem (CPU İşlemci) Birimi.....	5
3.1.4. Bilgisayar Veriyi Nasıl Saklar? .....	5
3.1.5. Çıkış Birimi .....	5
3.1.6. Program.....	5
3.2. Programlama Dili .....	5
3.3. Programlama Dillerinin Seviyelerine Göre Sınıflandırılması.....	6
3.3.1. Düşük Seviyeli Diller .....	6

3.3.2. Orta Seviyeli Diller.....	6
3.3.3. Yüksek(Üst) Seviyeli Diller .....	7
3.4. Programlama Dilinden Makine Diline Çevrim .....	7
3.5. İlk Bilgisayar Programcısı.....	7
<b>4. Algoritmalar .....</b>	<b>8</b>
4.1. Algoritmalar .....	8
4.2. Veri Türleri.....	8
4.2.1. Sayısal Veri .....	8
4.2.2. Karakter Verileri.....	8
4.2.3. Mantıksal Veri.....	9
4.3. Değişkenler ve Sabitler .....	10
4.3.1. Değişkenler .....	10
4.3.2. Sabitler .....	11
4.3.3. Değişkenlere isim verirken dikkat edilmesi gereken kurallar .....	11
4.4. Operatörler.....	12
4.4.1. Matematiksel Operatörler.....	12
4.4.2. Karşılaştırma Operatörleri (İlişkisel) Operatörler .....	13
4.4.3. Mantıksal Operatörler .....	13
4.4.4. İşlem Önceliği .....	14
4.5. Algoritmalar Nasıl Tasarlanır? .....	16
4.5.1. Hesaplama Kullanılacak Gereksinimlere Göre Tanımlamalar .....	16
4.5.2. Çözümün Hesaplanması İçin Yapılacak İşlemler.....	16
4.5.3. Hesaplamalara Göre Elde Edilen Sonuçların Kullanıcıya Sunulması .....	16
4.6. Algoritma Uygulamaları .....	16
4.6.1. Kahve Yapılışının Algoritması.....	16
4.6.2. Pasta Yapma Algoritması .....	16
4.6.3. Girilen İki Sayının Toplamını Yapan Algoritma .....	17
4.6.4. Dairenin Alanını Hesaplayan Algoritma .....	17
4.6.5. Karşılaştırma Operatörlerini Kullanan Algoritma Örneği .....	17
4.6.6. Aritmetik Ortalama Hesabı Yapan Algoritma Örneği .....	17
4.7. Akış Şemaları .....	18
4.8. Karar Yapıları.....	19
4.8.1. Karar Yapısı Örneği.....	19
4.9. Döngüler.....	20
4.9.1. Sayaçlar .....	20

4.9.2. While Döngüsü .....	21
4.9.3. Otomatik Sayaç Döngüsü(For) .....	22
4.9.4. İç İççe Döngüler .....	23
4.10. Algoritma ve Akış Şeması Örnekleri.....	24
4.10.1. Girilen İki Sayının Ortalamasını Hesaplama .....	24
4.10.2. Girilen İki Ders Notuna Göre Geçti Kaldı Hesabı .....	24
4.10.3. Sınıf Not Ortalaması .....	24
4.10.4. Ders Notu 50 ve Üzeri Olan Erkek Öğrencilerin Sayısı .....	25
4.10.5. Girilen Sayının Asal Sayı Olup Olmadığının Kontrolü .....	25
4.10.6. Faktöriyel Hesabı.....	26
4.10.7. Permütasyon Hesabı .....	26
4.10.8. Fibonacci Sayıları.....	27
4.10.9. İkinci Dereceden Denklemlerin Köklerini Bulma .....	27
4.10.10. Mükemmel Sayı Kontrolü .....	28
5. Çözümün Programlanması/Kodlanması.....	28
Kaynakça.....	A

# I. Bölüm: Algoritmalar

## 1. Etik, Güvenlik ve Toplum

### 1.1. Etik Değer

**Etik;** bireylerin ahlaklı ve erdemli bir hayat yaşayabilmesi için hangi davranışlarının doğru, hangilerinin yanlış olduğunu araştıran bir felsefe dalıdır. Temelinde barındırdığı güzel ahlaklı, adaletli ve iyi insan olma özellikleri değişmese de zamana, bilimsel gelişmelere ve toplumun gereklerine göre etik kavramına yüklenen anlam değişebilmektedir. Bir konuya ya da belirli bir meslek dalına özgü etik davranışların tamamı **etik değerler** olarak tanımlanabilir. Uluslararası Bilgisayar Etik Enstitüsüne göre bilişim teknolojilerinin doğru bir şekilde kullanılabilmesi için aşağıda belirtilen 10 kurala uyulması gerekmektedir.

1. Bilişim teknolojilerini başkalarına zarar vermek için kullanmamalısınız.
2. Başkalarının bilişim teknolojisi aracılığı ile oluşturduğu çalışmaları karıştırmamalısınız.
3. Başkasına ait olan verileri incelememelisiniz.
4. Bilişim teknolojilerini hırsızlık yapmak için kullanmamalısınız.
5. Bilişim teknolojilerini yalancı şahitlik yapmak için kullanmamalısınız.
6. Lisanssız ya da kırılmış/kopyalanmış yazılımları kullanmamalısınız.
7. Başkalarının bilişim teknolojilerini izinsiz kullanmamalısınız.
8. Başkalarının bilişim teknolojileri aracılığı ile elde ettiği çalışmalarını kendinize mal etmemelisiniz.
9. Yazdığınız programların ya da tasarladığınız sistemlerin sonuçlarını göz önünde bulundurmalısınız.
10. Bilişim teknolojilerini her zaman saygı kuralları çerçevesinde kullanmalıdır.

### 1.2. Bilgi Güvenliği, Sayısal Dünyada Parola Yönetimi

Hayatımızın önemli bir bölümünü oluşturan internet birçok güvenlik tehdidini kendi içinde barındırmaktadır. İnternet üzerinde yapacağımız ufak bir hata hayatımızın sonuna kadar bize büyük sorunlar yaşatabilir. İnternet üzerinden yüklenen bilgiler ya çok zor silinir ya da hiç silinmez. Sosyal medya üzerinden yapılan paylaşımlar neredeyse silinemez özelliğindedir. Sosyal medya üzerinde yapacağınız paylaşımlar ileride size büyük sorunlar açabilir. Örneğin günümüzde ne kadar yanlış olsa da iş alımlarında hesaplarınız birçok firma ve kuruluş tarafından incelenir ve sosyal medya profilinize göre işe alınırsınız.

Kullanıcıların yaptığı en büyük hata girdikleri tüm web siteleri için aynı şifreyi kullanmaktır. Aynı şifreyi kullanmak ne gibi sorunlar açabilir diye düşünürsek şöyle bir örnek verilebilir: Geçtiğimiz yıllarda çok fazla web sitesi kırıldı ve milyonlarca kullanıcı hesabı etrafa saçıldı. Sizin kullandığınız bir web sitesinin bu şekilde kırıldığını düşünün. Kırılan web sitesine ait hesabınızın şifresi kötü niyetli kişilerin eline geçecektir. Bu kişiler kırılan web sitesi üzerinden kolaylıkla sizin mail hesaplarınızı ve sosyal medya hesaplarınızı aynı şifreyi kullandığınız için rahatlıkla ele geçirecektir. Bu durumun sizin hayatınızda ne gibi sorunlar açabileceğini bir düşünün. Bu sorunun çözümü için kullandığınız her web sitesi için farklı şifreler belirlenmelidir. Eğer farklı şifreler kullanırsanız üyesi olduğunuz bir web sitesi kırılrsa dahi sizin bilgileriniz ile üye olduğunuz başka web sitelerine giriş yapılamayacaktır.

Farklı şifreleri akılda tutmak imkansızdır. Farklı şifre kullanmak için iki yol tercih edilebilir. Birinci yol, en basit yol olan şifrelerinizi bir deftere not almaktır. Defter kullanımı iki riski içerir. Defteri kaybederseniz tüm şifreleriniz gider ve bu şifreler başkaları tarafından rahatlıkla kullanılabilir. İkinci yol; **Keepass, LastPass, 1password** gibi yazılımları kullanmaktır. Bu yazılımlardaki mantık tüm şifrelerinizi yüksek güvenli şifreli bir dosya üzerinde saklamaktır. Şifreleri sakladığınız dosya bir ana şifre ile açılır ve dosya üzerinde bulunan şifrelere erişirsiniz. Bu uygulamalarda kullandığınız dosya, dosyanın ana şifresi bilinmiyorsa başkasının eline geçse dahi açılmaz. En büyük risk ana şifrenizin ve dosyanızın başkasının eline geçmesidir.



## 2.2. Problem Çözme Teknikleri

Problem çözme tekniklerini içselleştirir ve düşünme sürecinizin bir parçası hâline getirebilirseniz herhangi bir problemin çözümü sizin için daha kolay olacaktır. Program yazmak çoğunlukla problemleri çözmektir. Problem çözme tekniklerini bilmek kolay ve iyi program yazmanızı sağlayacaktır.

### 2.2.1. Her Zaman Bir Planınız Olsun

Her zaman bir planınız olmalıdır. Denediğiniz yaklaşım ne olursa olsun (doğru ya da yanlış), fikir üretmemekten ve deneme yapmamaktan her zaman daha iyidir. Planlama yapmak aynı zamanda hedef belirlemek ve bu hedefe ulaşmak anlamına gelir. Plan yaparsanız çözüme yönelik adımlar attığınızı ve zamanı etkili biçimde kullandığınızı göreceksiniz. Her bir adımda kendinize güveniniz artacak ve çözüme bir adım daha yaklaşmış olacaksınız.

### 2.2.2. Problemi Tekrar İfade Edin

Problemi tekrar ifade etmek, göremediğimiz bir ayrıntıyı görmemizi ya da problemi daha kolay çözmek adına bir ipucu yakalamamızı sağlayabilir.

### 2.2.3. Problemi Küçük Parçalara Ayırın

Verilen problemi adımlara ya da bölümlere ayırmak, çözümü kolaylaştırır. Bir problemi iki bölüme ayırırsak çözümde yarı yarıya kolaylaşır. Örneğin, elinizde 100 kişisel dosya olduğunu ve bu dosyaların alfabetik sıraya göre dizilmesi gerektiğini düşünelim. Önce bir dosya alıp sonra diğer dosyayı bir öncekine göre yerleştirebilirsiniz. Bu işi 100 dosya için tekrarladığınızı düşünün. Peki ya önce 100 dosyayı A-F, G-M, N-S ve T-Z olacak biçimde 4 gruba ayırırsanız işlemi daha kolay yapmaz mıydık?

### 2.2.4. Önce Bildiklerinizden Yola Çıkın

Programlama yaparken öncelikle bildiklerimiz ile başlamalı ve sonra yeni çözümler arayışına girmeliyiz. Problemi küçük parçalara bölerek çözebildiğiniz parçadan başlayınız. Bu parçaları çözerken diğer parçalarla ilgili olarak aklınıza yeni fikirler geldiğini ve aynı zamanda kendinize olan güvenin arttığını göreceksiniz. Programlama süreci boyunca çoğu zaman çok iyi olduğunuz konular, zorlandığınız konular ve henüz öğrenmediğiniz konular olacaktır.

### 2.2.5. Problemi Basitleştirin

Çözmekte zorlandığınız bir problemle karşılaşırsanız problemin kapsamını daraltmayı deneyebilirsiniz. Bunun için koşulları azaltmayı, problemin kapsama alanını küçültmeyi düşünebilirsiniz. Temel amacınız problemi basitçe ifade etmeye çalışmak olmalıdır.

### 2.2.6. Benzerlikleri Arayın

Benzerlik kavramı, çözülmesi istenen problemle önceden çözölen problem arasındaki olası örtüşmeleri bulmaktır. Problemin belirli bir bölümü başka bir problemle benzerlik gösterebilir.

### 2.2.7. Deneme Yapın

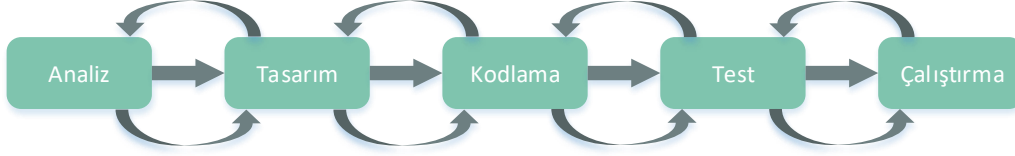
Bazen bir problemi çözmenin en kolay yolu denemek ve sonuçlarını gözlemlemektir. Böylece problemi çözebilmek için gereken ipuçlarını elde edebilirsiniz. Bir problemin çıktılarına bakarak sorunun nereden kaynaklandığını anlayabilir ve problemi çözebiliriz.

### 2.2.8. Asla Vazgeçmeyin

Program kodu çalışmadığı zaman programcı koda değil, kendisine ve aslında problemin kaynağına, yani kendi aklına kızmaktadır. Bu noktada moraliniz bozulursa başarısız olmak için bir bahane üretmiş oluruz. Böyle durumlarda en etkili çözüm ara vermektir. Problemden tamamen uzaklaşarak geçirilecek vakit sonrasında çok daha verimli çalışmanızı sağlar.

## 2.3. Yazılım Geliştirme Süreci

Yazılım geliştirilirken bir programcı ve yazılım gurubunun takip edeceği adımlar şu şekildedir. Altteki şekilden anlaşılacağı gibi adımlardan birinde bir sorunla karşılaşırsa bu sorunu çözebilmek için bir önceki adıma geri dönmek gerekecektir. Bu geri dönüş bazen birkaç adım olabilir.



Şekil 2 Yazılım Geliştirme Süreci

### 2.3.1. Analiz Aşaması

Analiz aşamasında yazılım projesinin çözeceği problemin tanımı yapılmaya çalışılır. Gereksinimler analiz edilir.

### 2.3.2. Tasarım Aşaması

Tasarım aşamasında tanımlanmış projenin hangi yol, yöntem, teknoloji v.b. ile çözüleceği belirlenir. Bu aşama problemin çözüm yolunun belirlendiği aşamadır.

### 2.3.3. Kodlama(Geliştirme) Aşaması

Kodlama (geliştirme) aşamasında tasarımda belirlenen çözüm uygulanır. Gerekli kodlar yazılır.

### 2.3.4. Test Aşaması

Test aşamasında geliştirilen çözüm, analizde belirlenen projenin gereksinimlerini karşılıyor mu diye bakılır.

### 2.3.5. Çalıştırma (Devreye Alma) Aşaması

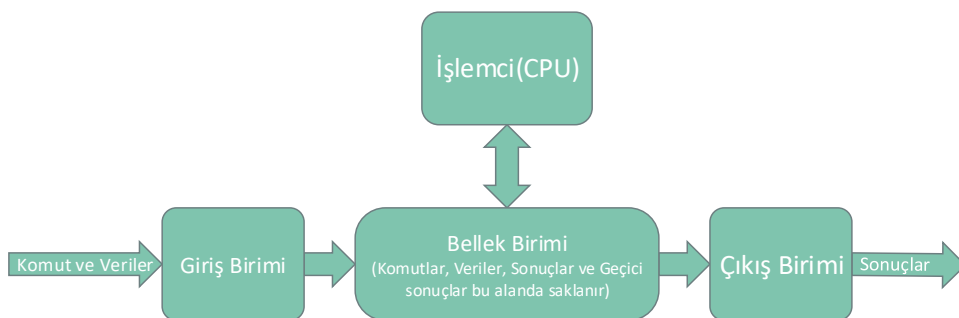
Çalıştırma (devreye alma) aşamasında geliştirilen yazılım çalıştırılır.

## 3. Bilgisayarın Çalışma Mantığı

### 3.1. Bilgisayar Nedir?

Bilgisayar, kendisine verilen bilgiler üzerinden aritmetiksel, mantıksal ve karşılaştırma işlemleri yaparak sonuçları çıktı birimlerine gönderen elektronik bir cihazdır. Bilgisayar çok yüksek hızlarda, bıkmadan usanmadan tekrarlı olarak aynı işleri yapan bir cihazdır.

Bilgisayarda *giriş, çıkış işlemleri, veri işleme ve veri saklama* işlemleri gerçekleştirilir. Kullanıcı tarafından girilen ham veriler, bilgisayar tarafından “girdi” olarak algılanır ve bu girdiler programlar tarafında işlenir. Kullanıcıya geri dönen değer, işlenmiş veridir. Bu veri “çıktı” olarak adlandırılır.



Şekil 3 Bilgisayarın Çalışma Mantığı



### 3.1.1. Giriş Birimi

Veri ve komutların bilgisayara yollanmasını sağlayan klavye, fare, mikrofon, kamera vb. cihazlardır.

### 3.1.2. Bellek Birimi

Veri, komut ve programların saklandığı donanımları temsil eder. Ana Bellek çoğunlukla RAM olarak anılır. Sabit disk yardımcı bellek olarak kullanılır ve bilgilerin kalıcı olarak saklanmasını sağlar.

**RAM** (Random Access Memory) çalışan uygulamaların içerisindeki anlık bilgileri tutan temel bir donanım parçasıdır. RAM'in hafıza büyüklüğü aynı anda çalışabilecek program sayısını artırır.

### 3.1.3. İşlem (CPU İşlemci) Birimi

Bilgisayardaki tüm Aritmetik ve Mantıksal işlemlerin yapıldığı birimdir. Tüm işlemleri yapan ana birimdir. Bilgisayar programının çalışmasını denetleyen, programda yapılan hesaplamalara göre hangi işlerin yapılacağına karar veren birimdir. Makine diline çevrilmiş bir programda komutları teker teker ve sırayla yorumlar ve çalıştırır.

### 3.1.4. Bilgisayar Veriyi Nasıl Saklar?

Bilgisayar veriyi hafızada saklar. Her bir veri için hafızada belirli bir alan ayrılır ve bu alan her seferinde tek bir değer saklayabilir. Kullanıcı, var olan değer yerine yeni bir değer atadığında eski değer silinir. Programın çalışması bittiğinde ya da bilgisayar kapatıldığında bu veriler silinir. Verilerin daha sonra tekrar kullanılması gerekiyorsa sabit diske kaydedilmeleri gerekir. Bu şekilde kaydedilen verilere “**dosya**” adı verilir. Temel anlamda *program dosyaları* ve *veri dosyaları* olmak üzere iki dosya türü vardır. Program dosyaları, bilgisayarın yapması istenen komutları ve işlemleri içerir. Veri dosyaları ise programlar çalışırken gereken verileri kapsar.

### 3.1.5. Çıkış Birimi

Bilgisayarda oluşan verilerin dış ortama verilmesini sağlayan ekran, yazıcı gibi birimlerdir.

### 3.1.6. Program

Program, bilgisayarlara yapması gerekenleri söyleyen, kullanıcılarının kullandığı, sabit diskte çalıştırılan verilerdir. İnternette gezinme, oyun oynama, mesajlaşma gibi işler programlar ile yapılmaktadır.

## 3.2. Programlama Dili

Bilgisayarlar; iki ana kısımdan oluşur **Yazılım** ve **Donanım**. Bilgisayarların yazılım kısmı donanımlara çeşitli elektronik sinyaller göndererek istenilen komutları gerçekleştirir. Programlama dilleri kullanıcıların kullandıkları yazılımları hazırlamaya yarayan programlardır. Programlama dilleri, yazılımcının bilgisayara neyi nasıl yapacağını hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar. *Özetle, yazılımlar programlama dilleri kullanılarak oluşturulurlar.*

Programlama dili kullanılarak yapılan işe **programlama** denir. Programlama dili ile yazılan metinlere **kod** denir.

Bilgisayarlar elektronik cihazlar oldukları için sadece ikilik düzeninde birler ve sıfırlar ile çalışır. Ancak birler ve sıfırlar ile programlama yapmak çok zordur. Günümüzde programlama dilleri programlamanın daha rahat bir şekilde yapılabilmesi için; konuşma dillerine yakın bir hale getirilmiştir. Her programlama dili diğerinden farklıdır. Her programlama dilinin bir diğerine göre üstün ve zayıf tarafları olduğundan en iyi programlama dili budur diye bir şey söylemeyiz.

*Günümüzde en yaygın kullanılan programlama dilleri şunlardır:*

*Java, C, C++, C#, Python, PHP, JavaScript, Swift...*

*Programlama hem problem çözme becerisi hem de bilgi işlemsel düşünme becerisine sahip olmayı gerektirir.*

### 3.3. Programlama Dillerinin Seviyelerine Göre Sınıflandırılması

Bir programlama dili konuştuğumuz doğal dile ne kadar yakın ise o kadar **yüksek seviyeli dil**, makine diline ne kadar yakın ise o kadar **düşük seviyeli dil** olarak sınıflandırılır.

#### 3.3.1. Düşük Seviyeli Diller

Bilgisayar donanımına doğrudan erişimi olan yazılması ve kontrolü çok zor olan dillerdir.

##### 3.3.1.1. Makine Dili (Birinci Seviye)

Makine dili, geliştirilen ilk programlama dilidir, öğrenilmesi çok zordur. Bu dilde yazılan tüm komutlar 0 ve 1'lerden oluşur. Belirli bir işlemci/makine için yazılan kod, farklı yapıdaki başka bir makinede çalışmaz, tamamen yeniden yazılması gerekir. Örneğin 1011101100010001 farklı yapıdaki işlemcilerde farklı işlemleri yapar. Bu yüzden makine dili donanıma bağlıdır. Makine dili biraz daha kolay okunabilmesi için 16'lık sayı sistemi ile yazılır fakat derlenme sonrası kod ikilik sisteme çevrilir. İşlemci ikilik tabandaki kodu okur ve uygular.

Özetle bilgisayarın ana dili ikili (binary) kodudur. İkili kod 0'lar ve 1'ler den oluşur.

**Makine Dili Örneği: Bu program ekrana "Hello world" yazısını yazar.**

```
1. 1011101100010001 0000000110111001 0000110100000000 1011010000001110
2. 1000101000000111 0100001111001101 0001000011100010 1111100111001101
3. 0010000001001000 0110010101101100 0110110001101111 0010110000100000
4. 0101011101101111 0111001001101100 0110010000100001
```

**Bu program ekrana "Hello world" yazısını yazar.**

```
1. BB11 01B9 0D00 B40E 8A07 43CD 10E2 F9CD 2048 656C 6C6F 2C20 576F 726C 6421
```

##### 3.3.1.2. Assembly Dili (ikinci seviye)

Bu dillerde komutlar sembollerle ifade edilir, makine koduna oldukça yakındırlar. Yüksek seviyeli dillerin yaygın kullanımına karşın, halen hızın ve verimin önemli olduğu noktalarda yazılan çoğu programda assembly dili kullanılmaktadır. Makine diline birebir çevrilebilir. Makine dilinden daha kolay anlaşılabilir (ama çok da değil).

**Assembly Programlama Dili Örneği: Bu program ekrana "Hello world" yazısını yazar.**

```
1. 10 format mz
2. 20 org 100h
3. 30 mov ah,09
4. 40 mov dx,yazi
5. 50 int 21h
6. 60 mov ah,00
7. 70 int 16h
8. 80 int 20h
9. 90 yazi db "merhaba dünya$"
```

#### 3.3.2. Orta Seviyeli Diller

Orta seviye dillerde yazılan programlar donanımdan bağımsız çalışırlar. Yani yazılan programlar farklı bilgisayarlarda kullanılabilirler. Oldukça esnek olan bu diller hem üst hem alt seviye programlama yapabilirler. Yüksek seviye dillerine göre kullanımı daha zordur fakat programcıya daha özgür bir program geliştirme imkânı sunarlar. C ve C++ dilleri orta seviyeli dillerdir.

**C programlama dili örneği: Bu program ekrana "merhaba, dünya" yazısını yazar.**

```
1. #include <stdio.h>
2. int main(void){
3.     printf("merhaba, dünya");
4.     return 0;
5. }
```

### 3.3.3. Yüksek(Üst) Seviyeli Diller

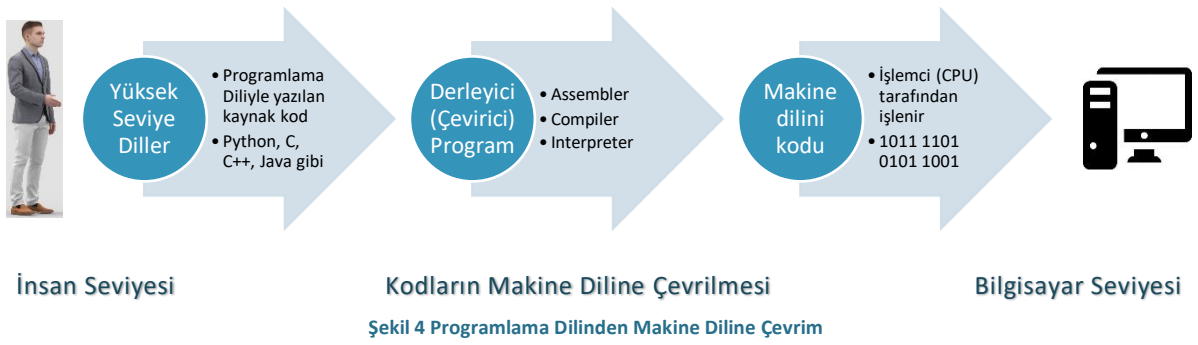
Yüksek seviyeli dilde, Assembly dili veya makine dilindeki birçok satır tek bir komutla gösterilir. Böylece program daha kısa bir sürede yazılır. Bu programlama dilleri programlama hâkimiyetini azaltırlar, bunun yanında en hızlı ve en etkili programlama dilleri bu kategoridedir. Yüksek seviyeli dilleri kullanmak ve öğrenmek kolaydır.

*Python programlama dili örneği: Bu program ekrana "merhaba, dünya" yazısını yazar.*

```
print("Merhaba Dünya")
```

### 3.4. Programlama Dilinden Makine Diline Çevrim

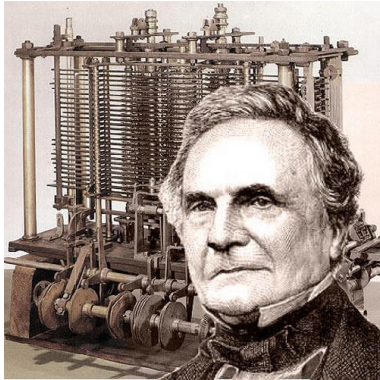
Programlama dillerinde yazılan programın çalışabilmesi için makine diline çevrilmesi gerekir. Programlama dillerinin kodlarını makine dilini çeviren **derleyici** denen programları vardır. Programlama diliyle yazılan kodlar derleyici aracılığıyla makine diline çevrilir. Derlenmiş olan programın içeriğine müdahale etme imkânı yoktur.



Şekil 4 Programlama Dilinden Makine Diline Çevrim

### 3.5. İlk Bilgisayar Programcısı

1837 **Charles Babbage** hesaplama yapabilen **Analitik Makine** adını verdiği bir cihaz tasarladı. Babbage tasarladığı Analitik makineyi zamanın teknolojisi yeterli gelmediği için hayata geçiremedi. Analitik Makine her ne kadar yapılamadıysa dahi zamanını birçok bilimi adamını etkilemişti.



Şekil 5 Charles Babbage Analitik Makine



Şekil 6 Ada Lovelace İlk Bilgisayar Programcısı

**Ada Lovelace** sahip olduğu maddi kaynak sayesinde Babbage'in en büyük destekçilerinden biriydi. Kendisi de matematikçi olan Ada Analitik Makine üzerine yazdığı notlarda Bernoulli sayılarını hesaplamak için Analitik Motorun Algoritmasını tanımlıyordu. İlk algoritmayı tasarladığı için **Ada ilk bilgisayar programcısı olarak kabul edilir.**

ABD Savunma Bakanlığı yararına yaratılmış bilgisayar dili **Ada**, Ada Lovelace isminden gelmiştir. **Ada Lovelace Günü** ocağın ortasında kutlanan yıllık etkinliktir ve amacı bilimdeki, teknolojiye, mühendislikteki ve matematikteki kadın profilini arttırmaktır.

## 4. Algoritmalar

### 4.1. Algoritmalar

**Algoritma;** bir problemin çözümünde uygulanacak işlemlerin maddeler halinde ya da şekiller yardımıyla sırası ile ifade edilmesidir. Algoritma günlük hayatta yapacağımız işler için hazırladığımız planların, programlama dillerindeki karşılığıdır. Diğer bir deyişle algoritma; çözeceğimiz ve uygulamaya döküleceğimiz problem için hazırlanacak işlem basamaklarıdır.

Bir problemi çözmek için algoritma adımlarını çıkarmadan, bu bilgiyi uygulamaya dökmek oldukça güç ve zaman kaybettirecek bir iştir. Programlama yapmak isteyen bilgisayar kullanıcılarının, herhangi bir programlama dilinden önce öğrenmesi gereken ilk şey; kesinlikle algoritma tasarlama teknikleri olmalıdır. Nasıl çözülebileceğine dair algoritması tasarlanmış bir problem, istenilen her programlama diliyle çözülebilir ve programlanabilir. Algoritma mantığını anlamış olmanız, herhangi bir programlama dilini öğrenmeniz için yeterlidir. Özetle: *Algoritma, bir sorunu çözebilmek için gerekli olan sıralı mantıksal adımlardır.*

### 4.2. Veri Türleri

Bilgisayara hangi veri türüyle çalışıyor olduğu mutlaka belirtilmelidir. Bir programda farklı veri türleriyle işlem yapılabilir. Tam sayılar, kesirli sayılar, karakterler, simgeler, metinler ve mantıksal değer, veri türlerini oluşturur.

#### 4.2.1. Sayısal Veri

Tüm sayı değerlerini içeren sayısal veri, hesaplama işlemlerinde kullanılabilen tek veri türüdür. Pozitif ya da negatif tam sayılar (3, 5, -7, -1023, 66578, 1981) ve ondalık (-56.23, 8695.235, 0.005, 3.14) sayılar kullanılabilir. Sayısal veriler; açılar, uzaklık, nüfus, ücret, yarıçap gibi hesaplama sürecinde gerekli değerler için tanımlanır.

#### 4.2.2. Karakter Verileri

Bilgisayar sayısal işlem yapabilen bir cihazdır. Bilgisayar her karaktere karşılık gelen bir sayı tutar ve işlemleri bu sayılar üzerinden gerçekleştirir. **ASCII** (American Standard Code for Information Interchange) olarak adlandırılan karakter seti bilgisayarın temel karakter setidir. Tablo 2 ASCII Tablosu Örneği üzerinde görüldüğü gibi "a" ve "A" gibi büyük ve küçük harfler farklı sayısal değerlere sahip olduğu için farklı algılanır. Karakter verileri karşılaştırılabilir ve alfabetik sıraya göre sıralanabilir. Karakterler kod içerisinde tırnak ("" ) içinde belirtilir. Harf ("a".."z", "A".."Z"), rakam ("0".."9") ve özel ("#", "&", "\*", ...) karakterleri karakter verileridir.

Karakterler sadece rakamlardan oluşsa bile hesaplama işlemlerinde kullanılamazlar.

#### Sayı Tipindeki Karakter Verileriyle İşlem Yapmak:

"3" + "5" = "35" İşleminin sonucu 8 olması gerekirken bu işlem karakterler ile yapılsaydı programlama dili ya hata verecekti ya da "35" değeri oluşacaktı

#### Sayı Verilerini ve Karakterleri Sıralamak

Sıra No	Sayı	Karakter
1	1	"1"
2	2	"10"
3	10	"1000"
4	11	"1010"
5	102	"102"
6	103	"103"
7	1000	"11"
8	1010	"2"

Bu örnekte sayıların ve karakterlerin bilgisayar içerisinde nasıl sıralandığı görülmekte. Eğer sayıları sıralarsanız numara sırasına göre sıralanacaktır. Fakat karakter olan sayıları sıralamak istediğinizde bilgisayar karakterleri alfabetik sıraya sokacaktır.

Karakter sütunu incelendiğinde bilgisayar sıralama "1" ile başlayanlara göre sıralamıştır. Bu yüzden "2" karakteri sütunun en sonundadır.

Tablo 2 ASCII Tablosu Örneği

Dec	Hex	Binary	Krktr	Dec	Hex	Binary	Krktr	Dec	Hex	Binary	Krktr	Dec	Hex	Binary	Krktr
32	20	00100000	Sp	56	38	00111000	8	80	50	01010000	P	104	68	01101000	h
33	21	00100001	!	57	39	00111001	9	81	51	01010001	Q	105	69	01101001	i
34	22	00100010	"	58	3A	00111010	:	82	52	01010010	R	106	6A	01101010	j
35	23	00100011	#	59	3B	00111011	;	83	53	01010011	S	107	6B	01101011	k
36	24	00100100	\$	60	3C	00111100	<	84	54	01010100	T	108	6C	01101100	l
37	25	00100101	%	61	3D	00111101	=	85	55	01010101	U	109	6D	01101101	m
38	26	00100110	&	62	3E	00111110	>	86	56	01010110	V	110	6E	01101110	n
39	27	00100111	'	63	3F	00111111	?	87	57	01010111	W	111	6F	01101111	o
40	28	00101000	(	64	40	01000000	@	88	58	01011000	X	112	70	01110000	p
41	29	00101001	)	65	41	01000001	A	89	59	01011001	Y	113	71	01110001	q
42	2A	00101010	*	66	42	01000010	B	90	5A	01011010	Z	114	72	01110010	r
43	2B	00101011	+	67	43	01000011	C	91	5B	01011011	[	115	73	01110011	s
44	2C	00101100	,	68	44	01000100	D	92	5C	01011100	\	116	74	01110100	t
45	2D	00101101	-	69	45	01000101	E	93	5D	01011101	]	117	75	01110101	u
46	2E	00101110	.	70	46	01000110	F	94	5E	01011110	^	118	76	01110110	v
47	2F	00101111	/	71	47	01000111	G	95	5F	01011111	_	119	77	01110111	w
48	30	00110000	0	72	48	01001000	H	96	60	01100000	`	120	78	01111000	x
49	31	00110001	1	73	49	01001001	I	97	61	01100001	a	121	79	01111001	y
50	32	00110010	2	74	4A	01001010	J	98	62	01100010	b	122	7A	01111010	z
51	33	00110011	3	75	4B	01001011	K	99	63	01100011	c	123	7B	01111011	{
52	34	00110100	4	76	4C	01001100	L	100	64	01100100	d	124	7C	01111100	
53	35	00110101	5	77	4D	01001101	M	101	65	01100101	e	125	7D	01111101	}
54	36	00110110	6	78	4E	01001110	N	102	66	01100110	f	126	7E	01111110	~
55	37	00110111	7	79	4F	01001111	O	103	67	01100111	g	127	7F	01111111	DEL

Tablo 3 Karakter Verileri

Veri Türü	Veri Seti	Örnek
Karakter	Tüm harfler, özel semboller ve rakamlar	"A", "Y", "K", "i", "6", "0", "+", "%"
Dizi	Birden fazla karakterden oluşan kombinasyon	"Bilgisayar", "F@tih", "Lule_burgaz" "532-5556633"

#### 4.2.3. Mantıksal Veri

Mantıksal veri, veri setinde yalnızca iki kelime barındırır: doğru ve yanlış. Bu veri evet ya da hayır şeklindeki karar verme süreçlerinde kullanılır. Örneğin elde edilen değer, beklenen değer mi, evli mi, arabası var mı, öğrenci lise mezunu mu gibi sonucu kesin doğru ya da yanlış olan durumlarda mantıksal veri tanımlaması yapılır.

Tablo 4 Veri Türleri Örnekleri

Veri	Veri Türü	Açıklama
Ürün satış bedeli: 49.99, 101.50	<b>Sayısal:</b> Reel	Bir ürünün satış bedeli hesaplama işlemlerinde kullanılır.
T.C. Kimlik No.: 10654876542	Karakter dizisi	Kimlik bilgileri hesaplama amaçlı kullanılmaz.
Ağırlık: 67	<b>Sayısal:</b> Tam sayı	Kilo cinsinden tam sayı olabilir ve hesaplamalarda kullanılır.
Şirket İsmi: ABC Firması	Karakter dizisi	Tamamen karakterlerden oluşur.
Kredi Onayı: Var, Yok	Mantıksal	Bu durumda onay ya vardır "Doğru" ya da yoktur "Yanlış".
Posta Kodu: 06110, 34217	Karakter dizisi	Posta kodları işlem yapmak için kullanılmaz.
Tarih: 21042017	Karakter dizisi, Sayısal Tam sayı	İşlem yapmak için tam sayı biçiminde tanımlanabilir; aksi takdirde dizi olarak tanımlanması daha uygundur.
IBAN: TR0600006543000012	Karakter dizisi	Para transferi için bankaya verilen kodlar hesaplama amaçlı kullanılmaz.

### 4.3. Değişkenler ve Sabitler

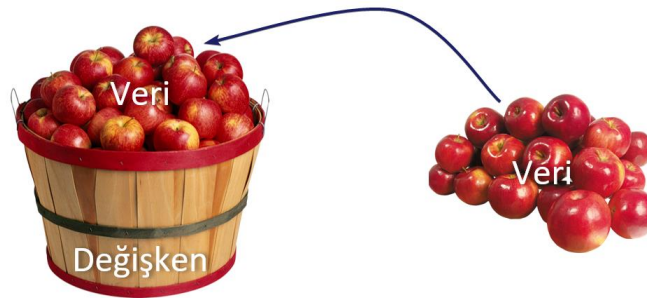
Bilgisayarlar problemleri çözmek için süreç boyunca sabit ve değişken olarak adlandırılan verileri kullanır.

#### 4.3.1. Değişkenler

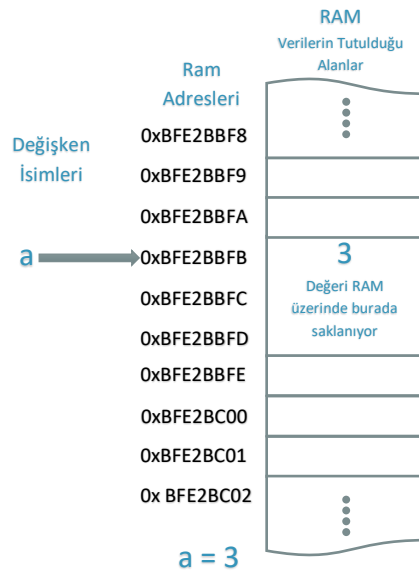
Programcı tarafından tanımlanan değişkenler, algoritmanın sonraki adımlarında kullanacağımız kendisine yüklenen verileri taşır. Değişkenlerin taşıdıkları değer kod içerisinde değiştirilebilir.

Adından da anlaşıldığı gibi değişkenlerin değerleri programın diğer adımlarında değiştirebilir ve içerisinde tutulan bilgilere ulaşabiliriz.

Bir değişkenin tanımlanması halinde hakkındaki tüm bilgiler RAM (geçici hafıza)'de tutulur. Bundan dolayı gereksiz değişken tanımlamadan olabildiğince kaçınılmalıdır.



Şekil 8 Değişken veri ilişkisi



a değişkenine 3 değeri atandığında a değişkeni şekilde görüldüğü gibi RAM üzerine bir alanda oluşturularak saklanıyor.

Şekil 7 Değişkenlerin RAM üzerinde oluşturulması

#### Değişken kullanılarak çözülen matematik problemi

946877. 946875 – 946876. 946875 = ?

**Çözüm:** sayıları tek tek çarpmak veya 946875 sayısına göre ortak paranteze almak sürekli 6 hanesi sayıları yazmayı gerektiriyor. Bu durum hem hata olasılığını artıracak hem de bizi çok uğraştıracaktır.

$x = 946875$  dersek  $(x+2).x - (x+1).x \rightarrow x(x+2-x-1) \rightarrow x$

Sonuç : x çıkar yani  $x=946875$



#### 4.3.2. Sabitler

Sabit olarak tanımlanan veriler problemin çözüm süreci boyunca asla değişmeyen değerlerdir. Program çalıştığı sürece bu değer kendisine verilen isim ile çağrılır ve değeri asla değiştirilemez. Örneğin, **pi** değeri değişmeyen bir değer olacağı için sabit olarak tanımlanmalıdır.



Şekil 9 Değişken ve Sabitler

#### 4.3.3. Değişkenlere isim verirken dikkat edilmesi gereken kurallar

1. Değişkene içerdği değer ile tutarlı isimler veriniz.
2. Değişkenlere isim verirken boşluk kullanmayınız.
3. Değişken isimleri bir karakter ile başlamalıdır.
4. Değişken isimleri sayı ile başlayamaz.
5. Matematiksel semboller kullanmamaya dikkat ediniz.
6. Bazı platformlar desteklemediği için Türkçe karakter kullanımı tavsiye edilmez.
7. Programlama dillerinde kullanılan komut isimleri değişken olarak kullanılamaz. Çok bilinenleri; **if, for, while, else, do, int**, vb.
8. Değişken isimlendirmelerinde boşluk karakteri yerine alt çizgi ( **\_** ) karakteri kullanılabilir.
9. Değişken isimlendirmede genellikle küçük harfle başlanır ve ikinci bir kelime yazılacaksa ilk kelimenin hemen ardından büyük harfle devam edilir. Buna “**Camel Karakter**” kullanımı denir. Örnek: **tcKimlikNo**
10. **:",<,>/?|\\()!@#\$\$%^&\*~+.** sembolleri değişken ismi içinde kullanılamaz. (Sadece “**\_**” sembolü kullanılabilir).

Tablo 5 Doğru Yanlış Değişken İsimleri Örnekleri

Yanlış	Doğru
1sayı	sayı1
Okul No.	okulNo
Soru?	Soru
i@	i

## 4.4. Operatörler

Operatörler bilgisayara ne tür bir işlem (matematiksel, mantıksal vb.) olduğuna dair bilgi verir. Operatörler; matematiksel, mantıksal ve ilişkisel operatörler olarak sınıflandırılabilir.

**Eşittir Operatörü "=":** "=" simgesi ile kullanılır. Eşittir operatörü sağına ve soluna olmak üzere iki adet ifade alır. Eşittir operatörünün sağında kalan ifade(ler), solunda kalan ifadeye aktarılır. Eşittir operatörünün sağında kalan kısımda birden fazla ifade varsa, öncelikle ifadeler hesaplanır. Hesaplanan sonuç eşitliğin sol tarafına aktarılır. Eşitliklere "**atama ifadeleri**" de denir.

### Operatör Kullanım Örneği:

**toplam** = 3 + 5

Bu örnekte kullanılan terimleri tanımlayalım:

<b>toplam</b>	Değişken
<b>3 ve 5</b>	Veriler
<b>+</b>	Operatör
<b>3 + 5</b>	İfade
<b>toplam = 3 + 5</b>	Eşitlik
<b>=</b>	Eşittir "=": Operatörü 3 + 5 ifadesindeki işlemi yapar ve solunda bulunan toplam değişkenine atar.
<b>8</b>	Sonuç: Operatörün yaptığı işlem sonucunda oluşan değer. Bu değer = işlemi sonucunda toplam değişkeninde saklanacaktır

**İşlem Gruplama Operatörü Parantezler ()** aynı matematikte olduğu gibi işlemleri gruplamak için kullanılır. Hem kodun okunabilirliğini arttırmak hem de işlemlerde hatayı engellemek için işlem gruplama kullanılır. Gruplanmak istenen ifade parantez "("" içine alınır.

### 4.4.1. Matematiksel Operatörler

Matematiksel operatörler; değişkenler ve veri tipleri üzerinde toplama, çıkarma, bölme, çarpma, üs alma ve mod alma gibi işlemler yapmamıza olanak sağlarlar.

Tablo 6 Matematiksel Operatörler

Operatör	Bilgisayar Sembolü	Örnek	Sonuç
Toplama	+	6.7 + 2.1	8.8
Çıkarma	-	5.6 – 3.4	2.2
Çarpma	*	3 * 4	12
Bölme	/	40 / 8	5
Üs Alma	^ veya **	2 ** 3	8
Tam Sayı Bölmesi	//	22 // 7	3
Modül Alma	%	10 % 6	4



#### 4.4.2. Karşılaştırma Operatörleri (İlişkisel) Operatörler

İki değişken ya da iki veri arasında büyüklük, küçüklük ve eşitlik kontrolü yapan operatörlere karşılaştırma (ilişkisel) operatörler denir. Karşılaştırma operatörleri; sonuç olarak geriye mantıksal (**bool**) veri tipinde **1 (True)** veya **0 (False)** olarak bir bit değer döner. Buna göre koşul sağlanmışsa True(1), sağlanmamışsa False(0) değeri döner.

Tablo 7 İlişkisel Operatörler

Operatör	Bilgisayar Sembolü	Örnek	Sonuç
Eşit	==	6 == 8	False
Küçüktür	<	6 < 8	True
Büyüktür	>	6 > 8	False
Küçük veya eşittir	<=	6 <= 8	True
Büyük veya eşittir	>=	6 >= 8	False
Eşit değildir	!=	6 != 8	True

#### 4.4.3. Mantıksal Operatörler

Mantıksal operatörler genellikle birden fazla karşılaştırmanın bulunduğu durumlarda kullanılır.

*Örneğin; Bir sayının 10'dan büyük 20'den küçük olduğu mantıksal operatörler yardımıyla yapılır.*

*sayı = 12 için eğer sayı > 10 ve sayı < 20 işlemi True döner*

##### 4.4.3.1. VE (AND) Operatörü

VE operatörü için çarpım işlemi yapıyor gibi düşünülebilir.

Tablo 8 VE Operatörü

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Tablo 9 VE Operatörü Örneği

Girilen A Sayısı	A > 10	A < 20	(A > 10) and (A < 20)
5	0	1	0
15	1	1	1
25	1	0	0

#### 4.4.3.2. VEYA (OR) Operatörü

VEYA operatörü toplama işlemi yapıyor gibi düşünülebilir.

Tablo 10 VEYA Operatörü

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Tablo 11 Veya Operatörü Örneği

Girilen A Sayısı	A < 15	A > 25	(A < 15) or (A > 25)
10	1	0	1
20	0	0	0
30	0	1	1

#### 4.4.3.3. DEĞİL (NOT) Operatörü

Değil operatörü mantıksal değer tersini alır. Değer 1 (True) ise 0, 0 (False) ise 1 olur.

Tablo 12 Değil Operatörü

A	not A
0	1
1	0

#### 4.4.4. İşlem Önceliği

Matematiksel, mantıksal ve ilişkisel operatörlerin bir hiyerarşisi yani öncelikleri vardır. İşlemler, bu sıralamaya göre yapılmaz ise sonuç, beklendiği gibi çıkmayabilir. En içteki parantezden en dıştakine doğru işlem yapılmalı, parantez içerisinde ise işlem önceliklerine dikkat edilmelidir. Aşağıdaki tabloda operatörlerin işlem önceliği yukarıdan aşağıya doğru listelenmiştir.

*Kodlamadaki işlem önceliği hataları, yanlış sonuçlar alınmasına neden olabilir. Bu tür hatalar **mantık hataları** oldukları için uygulamada akışı engelleyen bir hata ile karşılaşılmaz. Bu nedenle mantık hatalarının nerede yapıldığının bulunması güçtür.*

Tablo 13 Operatör Önceliği

İşlem Sırası	Açıklama
()	Parantez içerisindeki işlemler en içten en dışa doğru yapılır.
f(args...)	Fonksiyonlar
**	Kuvveti (Üs) operatörü
*, /, %, //	Çarpma, bölme, mod
+, -	Toplama, çıkarma
==, <, >, <=, >=, !=	İlişkisel operatörler
NOT	Mantıksal operatörler
AND	Mantıksal operatörler
OR	Mantıksal operatörler

Tablo 14 İşlem Önceliği Örnekleri

Değişken	İşlem	İşlem	Çıktı
x = 10 y = 15 z = 20	x+y-z	10+15-20	5
	x-y*z	10-15*20	-290
	(x-y)*5	(10-15)*5	-25
	x<y AND x<z	10<15 ve 10<20	True
	x != y OR z >=20	10!=15 veya 20>=20	True

*x = 2, y = 4, z = 12 için*

$(10+x+y)/(z+y)*(y+x)/x$

$(10+2+4)/(12+4)*(4+2)/2$

$(16)/(16)*(6)/2$

6/2

3

$10+x+y/z+y*y+x/x$

$10+2+4/12+4*4+2/2$

$12+1/3+16+1$

29+0,33

29,33

## 4.5. Algoritmalar Nasıl Tasarlanır?

Algoritmalar **sıralı adımlardan** oluşurlar. Her bir adım alt alta yazılır ve yukarıdan aşağı doru bir sırayla ilerler. Algoritmanın iskeleti 3 adımdan oluşur;

### 4.5.1. Hesaplama Kullanılacak Gereksinimlere Göre Tanımlamalar

Problemin çözümüne dair yapılacak işlemler burada başlar. Problemin çözümünde yapılacak işlemler için kullanılacak unsurlar ve gereksinimlere göre bu adımda tanımlamalar yapılır. Bu adımda **Değişkenler** ve **Sabitler** kullanılır.

### 4.5.2. Çözümün Hesaplanması İçin Yapılacak İşlemler

Bu adımda problemin çözülmesi için gerekli hesaplamalar yapılır. Bu adımda yapılacak işlemler operatörler yardımıyla gerçekleştirilir.

### 4.5.3. Hesaplamalara Göre Elde Edilen Sonuçların Kullanıcıya Sunulması

Bu adımda problemin çözümünde elde edilen ve kullanıcıya gösterilmek çıktılar kullanıcıya sunulur.

## 4.6. Algoritma Uygulamaları

Algoritmalar konusunu en iyi anlamanın yolu algoritmalar ile ilgili örnekleri incelemek ve örnekleri kendi başımıza yapmaktır. Şimdiye kadar algoritmalar konusu başlığı altında gördüğümüz birçok bilgi teorik düzeyde verildi. Bu bölümde algoritmalar konusunun anlaşılması için uygulamalar yapılacak. Uygulamaları incelerken anlatılan teorik bilgileri önünüzde bulundurun ve takıldığınız yerlerde bu bilgilere bakın.

### 4.6.1. Kahve Yapılışının Algoritması

Kettle'a su koy.  
Kettle'ı prize tak  
Su kaynayınca kettle'ın prizini çek  
Suyu bardağa koy  
Bardağa iki kaşık nescafe koy  
İstiyorsan şeker ilave et  
karıştır

Bu ilk örneğimizde aklımıza gelen en basit uygulamanın algoritmasını hazırladık. İşlem basamaklarını okuduğunuz zaman hiç bilmiyorsanız dahi rahatlıkla nescafe hazırlayabilirsiniz. Bilgisayar için yazılan algoritmalarda bu örneğe çok benzemektedirler.

### 4.6.2. Pasta Yapma Algoritması

1. Pastanın yapımı için gerekli malzemeleri hazırla
2. Yağı bir kaba koy
3. Şekerini aynı kaba yağın üzerine koy
3. Yağ ve şekerini çırp
5. Karışımın üzerine yumurtayı kır
6. Tekrar çırp
7. Kıvama geldi mi diye kontrol et
8. Kıvamlı ise 9. adıma devam et  
Değilse 6. adıma dön.
9. Karışımına un koy
10. Karışımına vanilya, kabartma tozu vb. koy
11. Karışımı Kıvama gelinceye kadar çırp
12. Pastayı Kek kalıbına koy
13. Yeteri kadar ısınan fırına pastayı koy
14. Pişimi diye kontrol et
15. Pişmiş ise 16. adıma devam et  
Değilse 14. adıma dön
16. Keki fırından çıkart
17. Fırını kapat
18. Keki ı kapat
19. Keki soğumasını bekle
20. Keki servis edebilirsin.

Bu örnek günlük hayatımızda yaptığımız daha karmaşık bir işin algoritması olan pasta yapma algoritmasıdır. Bu algoritmanın önceki algoritmaya göre en büyük farkı içinde kontrol ifadeleri bulundurması. 8. ve 15. Adımda görüldüğü gibi gerekli kontroller sağlanana kadar işlem bir sonraki adıma geçmiyor.

#### 4.6.3. Girilen İki Sayının Toplamını Yapan Algoritma

1. Birinci sayıyı gir: birinci\_sayı
2. İkinci sayıyı gir: ikinci\_sayı
3. toplam = birinci\_sayı + ikinci\_sayı
4. toplam yazdır.

Bu algoritma bilgisayara yaptırılacak en basit algoritma ördeği olmasının yanında algoritma nasıl tasarlanır sorusunun cevabını verdiği için çok önemlidir.

1. ve 2. Adımda kullanılacak gereksinimler tanımlanır

3. adımda çözüm hesaplanır

4. adımda elde edilen sonuçlar kullanıcıya sunulur.

#### 4.6.4. Dairenin Alanını Hesaplayan Algoritma

$\pi$  sabiti oluştur:  $\pi = 3.14$   
r sayısını oku  
alan =  $\pi * (r * r)$   
alan yazdır

Bu algoritma örneğinde  $\pi$  sabitini tanımladık.  $\pi$  değeri gerçek hayatta olduğu gibi program içinde de değişmeyecektir.

Alan ve r değişkenleri kullanıcının gireceği değere göre değişebileceği için alan ve r için değişkenler kullanıldı.

#### 4.6.5. Karşılaştırma Operatörlerini Kullanan Algoritma Örneği

A sayısını klavyeden oku  
B sayısını klavyeden oku  
Eğer  $A == B$  ise Girilen sayılar birbirlerine eşittir  
Eğer  $A != B$  ise Girilen sayılar birbirlerinden farklıdır  
Eğer  $A > B$  ise A sayısı B sayısından büyüktür  
Eğer  $A >= B$  ise A sayısı B sayısından büyüktür ya da eşittir  
Eğer  $A < B$  ise A sayısı B sayısından küçüktür  
Eğer  $A <= B$  ise A sayısı B sayısından küçüktür ya da eşittir

Bu örnekte klavyeden girilen iki sayının birbiriyle karşılaştırması yapılıyor.

#### 4.6.6. Aritmetik Ortalama Hesabı Yapan Algoritma Örneği

1. x1, x2, x3, x4, x5 gir
2. toplam = 0
3. toplam = toplam + x1
4. toplam = toplam + x2
5. toplam = toplam + x3
6. toplam = toplam + x4
7. toplam = toplam + x5
8. ortalama = toplam / 5
9. ortalama yazdır






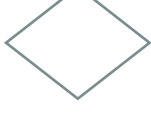


Bu örnekte klavyeden girilen 5 sayının aritmetik ortalaması bulunuyor.

3. adımdan 7. Adıma kadar yazılan basamaklardaki adımlara dikkat edin! Sizce bu adımları daha kolay yapmanın bir yolu olabilir mi?

## 4.7. Akış Şemaları

Problem çözme sürecimiz, bilgisayarın iletişim kurma yöntemi ile şekillenir. Algoritma, bilgisayara hangi işlemi hangi sırada yapması gerektiğini söyler. Akış şeması ise algoritmanın görsel gösterimidir. Programcı, oluşturulan algorithmadan grafiksel gösterimler oluşturur. Akış şeması, program geliştirmeye başlamadan önceki son adımdır. Akış şemasında hatalar rahatlıkla görülüp düzeltilebilir.

Akış şeması, bir problem çözümünün başlangıcından bitişine kadar olan süreci gösterir. Akış şeması içerisindeki her bir simge, algorithmadaki bir işlemi ifade eder.

Simge	İşlev	
	Akış Yönü	Algoritma tasarımında bir işlem bittikten sonra işleme nereden devam edilmesi gerektiğini gösteren şekildir. Algoritmanın anlaşılabilir olması için her bir işlem arasında varsa koşullarıyla birlikte bu şekil çizilir.
	Bağlaç	Akış yönlerinin bağlantısını sağlar.
	Giriş	Programa klavyeden veya diğer giriş birimlerinden bilgi alınacağını gösteren akış diyagramı şeklidir.
	Çıkış	Programda ekrandan ya da diğer çıkış birimlerinden çıktı alınması istenildiği zaman kullanılan akış diyagramı şeklidir.
	Atama/İşlem	Programda yapılması istenilen matematiksel işlemlerin ifade edildiği akış diyagramı şeklidir.
	Denetim (Karar)	Karşılaştırma işlemlerinde kullanılan akış diyagramı şeklidir. İçerisine yazılan koşul veya koşulların sağlanması ile sağlanmaması halinde neler yapılması gerektiğini belirlemede kullanılır.
	Döngü	Programda belli bir kod bloğunun, belirli bir tekrar kadar işlenmesi isteniyorsa kullanılan şekildir.
	Önceden Tanımlı İşlem/Fonksiyon	Program içerisinde kullanılan fonksiyonları gösteren şekildir.

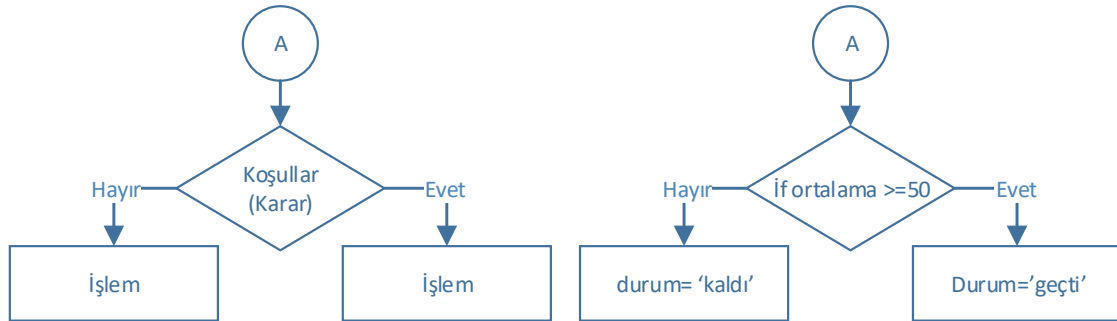
Şekil 10 Akış şeması sembolleri

Akış şemalarını oluştururken dikkat edilmesi gereken bazı noktalar şunlardır:

1. Yönergeler, simgelerin içine yazılmalıdır.
2. Hatırlatıcı bilgiler simgenin yanına yazılabilir. Böylece akış şeması ek açıklamalı bir şemaya dönüşür.
3. Bir akış şeması her zaman sayfanın başından başlar ve sonuna doğru gider. Eğer bir sayfaya sığmazsa bir ya da daha fazla bağlantı simgesi kullanılarak diğer sayfaya geçilebilir.
4. Akış şemasını çizmek için uygun yazılımlar kullanılırsa daha standart bir görünüm elde edilir.
5. Simgeler, içeriğindeki yazının rahatça okunabileceği kadar büyük yapılmalıdır.

## 4.8. Karar Yapıları

Karar yapıları bir algorithmada bir kararın verilmesini ve bu karara göre iki seçenektan birinin uygulanmasını sağlar. Eşkenar dörtgen içerisine kontrol edilecek mantıksal koşul yazılarak gösterilir. Program akışı sırasında koşulun doğru olması durumunda "Evet" yazılan kısma Yanlış olması durumunda "Hayır" yazılan kısma sapılır. Tek girişli ve çift çıkışlı bir şekildir. Akış şemalarında eğer kelimesinin İngilizce karşılığı olan "If" kelimesi kullanılacak.



Şekil 11 Karar Yapıları

### 4.8.1. Karar Yapısı Örneği

Algoritma	Akış Şeması
<p>Öğrencilerin dönem sonu not ortalamasına göre kaldı, geçti ve hangi belgeyi aldığını bulan algoritma ve akış şeması örneği.</p>	<pre>graph TD     A((A)) --&gt; Karar1{İf ortalama &gt;=50}     Karar1 -- Hayır --&gt; Durum1[Durum='Kaldı']     Karar1 -- Evet --&gt; Karar2{İf ortalama &gt;=85}     Karar2 -- Evet --&gt; Durum2[Durum='Takdir']     Karar2 -- Hayır --&gt; Karar3{İf ortalama &gt;=70}     Karar3 -- Evet --&gt; Durum3[Durum='Teşekkür']     Karar3 -- Hayır --&gt; Durum4[Durum='Belge Yok']</pre>

## 4.9. Döngüler

Programda belli bir kod bloğunun, belirli bir tekrar kadar işlenmesi gerekiyorsa döngü algoritmaları kullanılmalıdır.

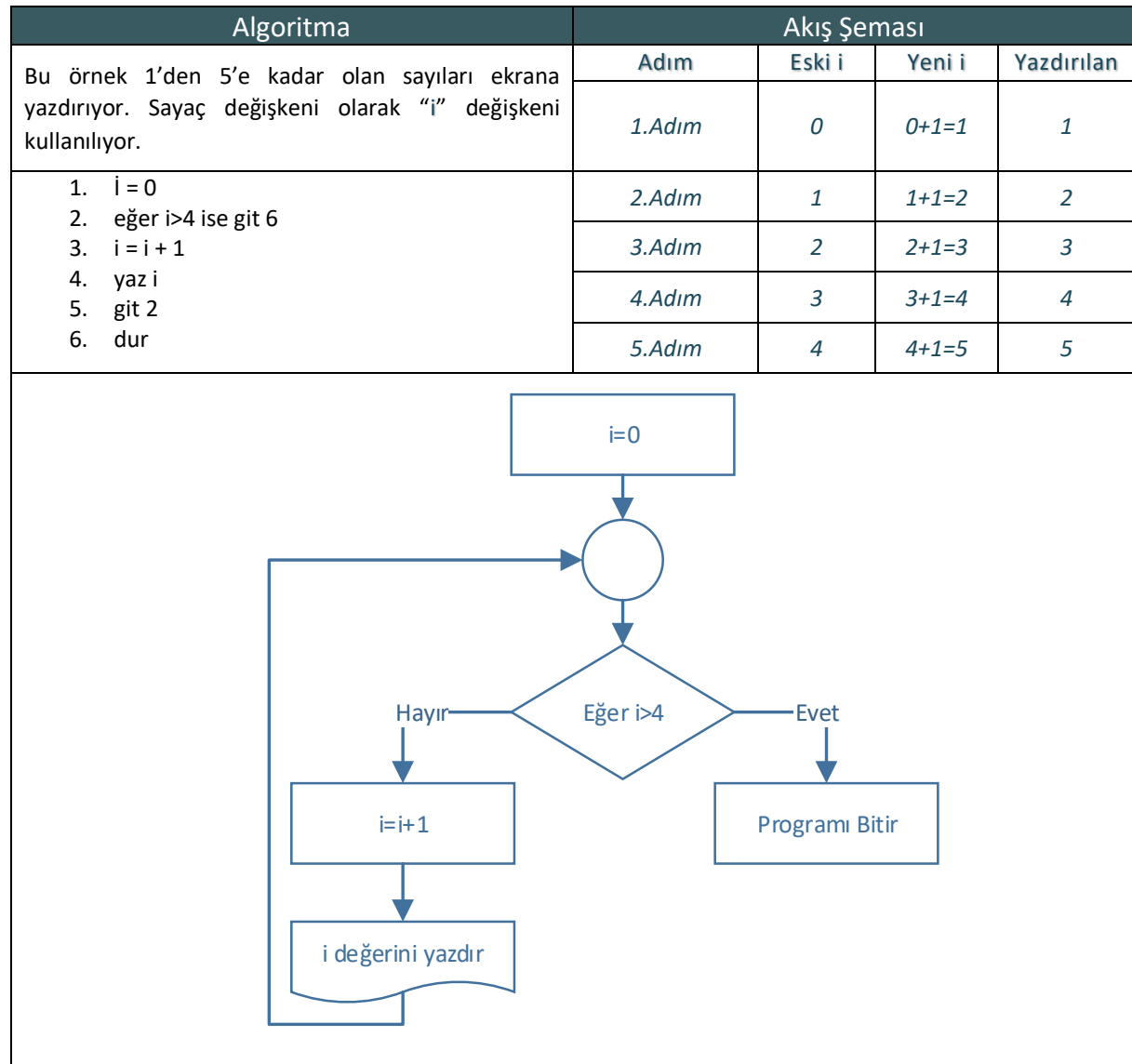
### 4.9.1. Sayaçlar

Programlarda bazı işlemlerin belirli sayıda yaptırılması veya işlenen/üretilen değerlerin sayılmasında kullanılan değişkenlere sayaç denir.

$sayaç = sayaç + 1$

Bu işlemde sağdaki ifadede değişkenin eski değerine 1 eklenmekte; bulunan sonuç yine kendisine yeni değer olarak aktarılmaktadır. Kodlamada sayaç olarak i ve j değişken isimleri kullanılır.

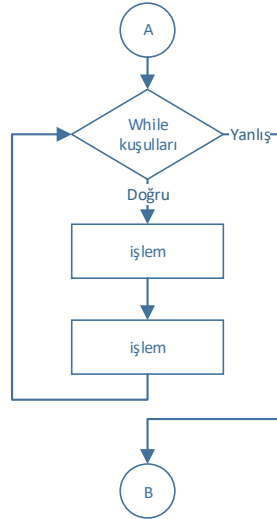
#### 4.9.1.1. Sayaç Örneği





#### 4.9.2. While Döngüsü

Bu döngü yapısı bilgisayara, koşul doğru olduğu sürece işlemleri tekrarlanmasını belirtir. Koşul daha çevrim içerisine girmeden sınanır. Koşul olumsuz olduğunda çevrime hiç girilmez ve döngü içerisinde yapılması gerekenler atlanır.



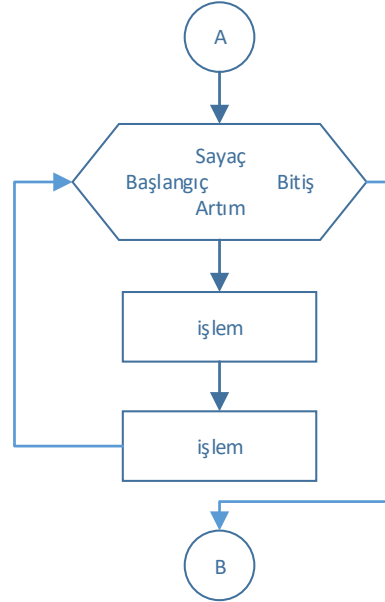
Şekil 12 While Döngü Yapısı

##### 4.9.2.1. While Döngüsü Örneği

Algoritma	Akış Şeması
<p>Bir sınıftaki öğrencilerin yaş ortalamasını hesaplayan algoritma ve akış şeması örneği;</p>	
<pre>tpYaş=0 tpÖğrn=0 öğrenci yaşını giriniz Yaş 0'a eşit değilse dön     tpYaş = tpYaş +yaş     tpÖğrn = tpÖğrn +1     öğrenci yaşını giriniz ortalama= tpYaş / tpÖğrn ortalamaı yazdır.</pre>	

#### 4.9.3. Otomatik Sayaç Döngüsü(For)

Bu yapıda döngünün her tekrarında bir değişkenin değeri arttırılır ya da azaltılır. Bu döngüyü tasarlayan programcı, döngü tekrar sayısını kontrol etmek için değişken olarak tanımlanan bir sayaç tutar. Döngü, bu sayaç bitirme sayısından büyük olana kadar tekrar eder.



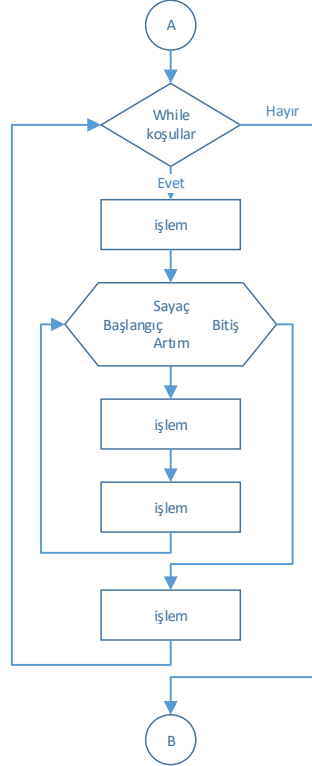
Şekil 13 5.7.3. Otomatik Sayaç Döngüsü(For)

##### 4.9.3.1. Otomatik Sayaç Döngüsü(For) Örneği

Algoritma	Akış Şeması
<p>Bir sınıfın yaş ortalamasını bulmak için otomatik sayaç döngü yapısına ait algoritma ve akış şeması</p>	
<p>KişiSayısı=12 toplam=0 j 1'den 12 olana kadar dön Yaşı Gir. toplam=toplam + yaş ortalama= toplam/j Ortalamayı yazdır, ortalama</p>	

#### 4.9.4. İç İçe Döngüler

İç içe döngüler, tekrarlayan bir dizi işlem yine tekrar edilmek istendiğinde kullanılır. İçteki döngülerin dıştaki döngülerle aynı döngü yapısında olması gerekmez. Dıştaki döngü **while** yapısında iken içteki döngü **for** yapısında olabilir. Ancak içteki ve dıştaki döngüyü kontrol eden değişkenin farklı olması gerekir.



Şekil 14 İç İçe Döngüler

##### 4.9.4.1. İç İçe Döngü Örneği

Algoritma	Akış Şeması
<p>Çarpım tablosunu ekrana yazdıran algoritma.</p> <pre>i=1 while i&lt;=10:     for j 1'den 10 a kadar dön         i, j ve i*j yazdır     i=i+1</pre>	<pre>graph TD     i1[i=1] --&gt; While1{While i&lt;=10}     While1 -- Evet --&gt; j1[j 1 10]     j1 --&gt; Print[i, j ve i*j yazdır]     Print --&gt; iplus[i=i+1]     iplus --&gt; While1     While1 -- Hayır --&gt; Cikis((Çıkış))</pre>

## 4.10. Algoritma ve Akış Şeması Örnekleri

### 4.10.1. Girilen İki Sayının Ortalamasını Hesaplama

Algoritma	Akış Şeması
<pre>not1 oku not2 oku ortalama = (not1 + not2) / 2 ortalama Yaz</pre>	<pre>graph TD     Input[/not1 gir not2 gir/] --&gt; Process[Ortalama=(not1+not2)/2]     Process --&gt; Output[/ortalama, yaz/]</pre>

### 4.10.2. Girilen İki Ders Notuna Göre Geçti Kaldı Hesabı

Algoritma	Akış Şeması
<pre>not1 oku not2 oku ortalama = (not1 + not2) / 2 eğer ortalama &gt;= 50 ise     "Geçer" yaz     Değilse "Kaldı" yaz</pre>	<pre>graph TD     Input[/not1 gir not2 gir/] --&gt; Process1[Ortalama=(not1+not2)/2]     Process1 --&gt; Output1[/ortalama, yaz/]     Output1 --&gt; Decision{Ortalama &gt;= 50}     Decision -- Evet --&gt; Output2[/Geçti/]     Decision -- Hayır --&gt; Output3[/Kaldı/]</pre>

### 4.10.3. Sınıf Not Ortalaması

Algoritma	Akış Şeması
<pre>sinif_mevcutdu gir toplamlam=0 for i=1 den sınıf_mevcutdu kadar dön     ogrnc_ders_notu gir     toplamlam = toplamlam + ders_notu ortalamlam=toplamlam/sinif_mevcutdu Sınıf Ortalaması, ortalamlam yazdır</pre>	<pre>graph TD     Input[/mevcut gir/] --&gt; Process1[toplamlam=0]     Process1 --&gt; LoopStart(( ))     LoopStart --&gt; Decision{1 J mevcut}     Decision --&gt; Input2[/ogmc_ders_notu, gir/]     Input2 --&gt; Process2[toplamlam=ogmc_ders_notu+toplamlam]     Process2 --&gt; LoopStart     LoopStart --&gt; Process3[ortalamlam=toplamlam/ mevcut]     Process3 --&gt; Output[/ortalamlam, yaz/]</pre>

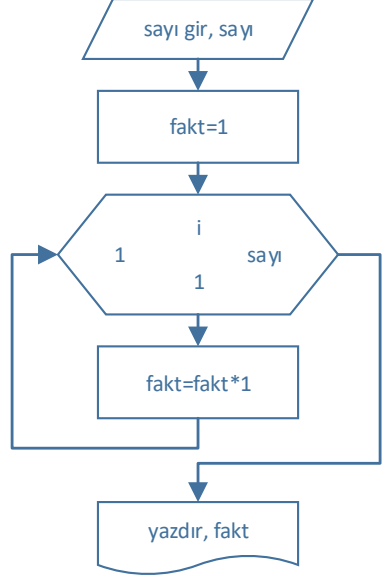
#### 4.10.4. Ders Notu 50 ve Üzeri Olan Erkek Öğrencilerin Sayısı

Algoritma	Akış Şeması
<pre> toplam=0 çıkış="" Çıkış için Ç girin yazdır while çıkış != "Ç"     Cinsiyeti Giriniz E/K gir, cinsiyet    Ders Notu gir,     ders_not     if cinsiyet=="E" and ders_not&gt;=50         toplam+=1     Çıkış için ç giriniz gir, çıkış Ders Notu 50 ve Üzeri Olan Erkek Öğrenci Sayısı: yazdır ,toplam         </pre>	<pre> graph TD     Start([toplam=0 çıkış=""]) --&gt; Input[/Çıkış için 'Ç' girin/]     Input --&gt; Loop{while çıkış != 'Ç'}     Loop -- Evet --&gt; InputCinsiyet[/cinsiyet girişi E/K, cinsiyet/]     InputCinsiyet --&gt; InputDersNotu[/ders notu girişi, ders_notu/]     InputDersNotu --&gt; Decision{if cinsiyet=='E' and ders_notu&gt;=50}     Decision -- Hayır --&gt; Loop     Decision -- Evet --&gt; Increment[toplam=toplam+1]     Increment --&gt; Loop     Loop -- Hayır --&gt; Output[/çıkış için ç giriniz, çıkış/]     Output --&gt; End([ders notu 50 ve üzeri olan erkek öğrencilerin sayısı, toplam])         </pre>

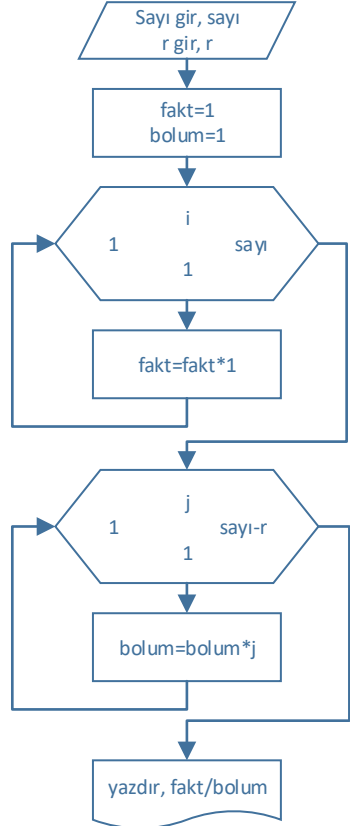
#### 4.10.5. Girilen Sayının Asal Sayı Olup Olmadığının Kontrolü

Algoritma	Akış Şeması
<p>Sadece 1 ve kendisine bölünen sayılara asal sayılar denir.</p> <pre> toplam=0 Sayı Girin, sayı for i=1'den sayı+1 e kadar dön     if sayı%i==0         toplam=toplam+1 if toplam==2     Asal, yaz else     Asal Değil, yaz         </pre>	<pre> graph TD     Start([toplam=0]) --&gt; Input[/sayı gir/]     Input --&gt; Loop{for i=1'den sayı+1 e kadar dön}     Loop --&gt; Decision{if sayı%i==0}     Decision -- Evet --&gt; Increment[toplam=toplam+1]     Decision -- Hayır --&gt; Loop     Increment --&gt; Loop     Loop --&gt; Decision2{if toplam==2}     Decision2 -- Evet --&gt; Output1[/Asal, yazdır/]     Decision2 -- Hayır --&gt; Output2[/Asal Değildir, yazdır/]         </pre>

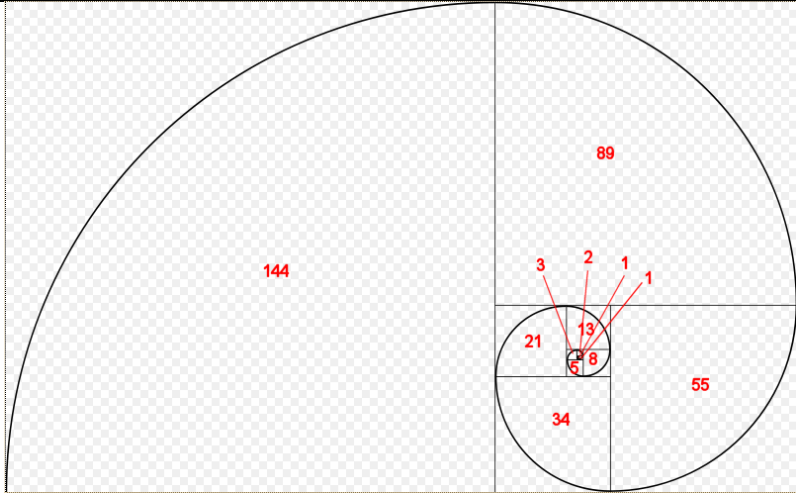
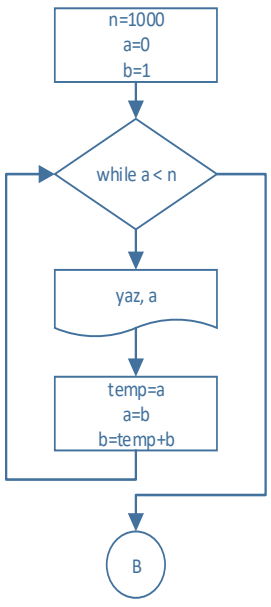
#### 4.10.6. Faktöriyel Hesabı

Algoritma	Akış Şeması
$n! = 1.2.3 \dots n$	 <pre> graph TD     Start[/sayı gir, sayı/] --&gt; Fakt1[fakt=1]     Fakt1 --&gt; Loop{i=1 sayı}     Loop --&gt; FaktMul[fakt=fakt*i]     FaktMul --&gt; Loop     Loop --&gt; End[/yazdır, fakt/] </pre>
<p>Sayı gir, sayı</p> <p>fakt=1</p> <p>for i=1 den sayı kadar dön fakt = fakt * i</p> <p>yazdır, faktöriyel</p>	

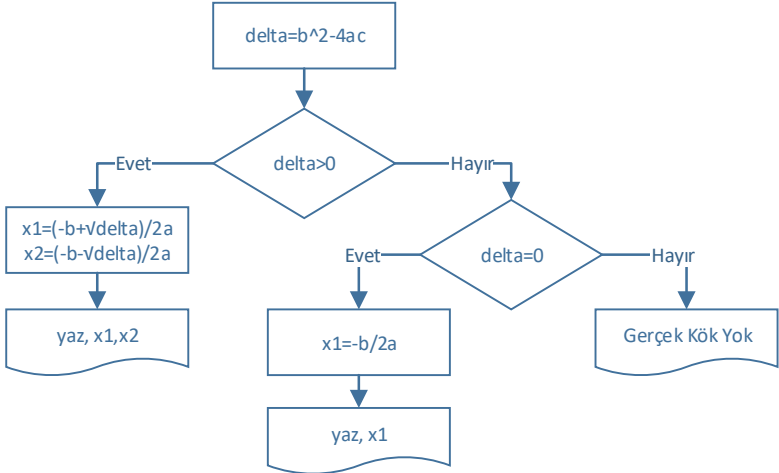
#### 4.10.7. Permütasyon Hesabı

Algoritma	Akış Şeması
$p(n,r) = \frac{n!}{(n-r)!}$	 <pre> graph TD     Start[/Sayı gir, sayı r gir, r/] --&gt; Init[fakt=1 bolum=1]     Init --&gt; Loop1{i=1 sayı}     Loop1 --&gt; FaktMul[fakt=fakt*i]     FaktMul --&gt; Loop1     Loop1 --&gt; Loop2{j=1 sayı-r}     Loop2 --&gt; BolumMul[bolum=bolum*j]     BolumMul --&gt; Loop2     Loop2 --&gt; End[/yazdır, fakt/bolum/] </pre>
<p>Sayı gir, sayı r gir, r</p> <p>fakt=1 bolum=1</p> <p>for i=1 den sayı kadar dön fakt = fakt * i</p> <p>for j=1 den (sayı -r) kadar dön bolum = bolum * j</p> <p>yazdır, fakt/bolum</p>	

#### 4.10.8. Fibonacci Sayıları

Algoritma	Akış Şeması
 <p>0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987</p> <pre> n=1000 a=0 b=1 while a &lt; n:     yaz, a     temp=a     a=b     b=temp+b </pre>	 <pre> graph TD     Start([n=1000 a=0 b=1]) --&gt; Loop{while a &lt; n}     Loop --&gt; Print[yaz, a]     Print --&gt; Update[temp=a a=b b=temp+b]     Update --&gt; Loop     Loop --&gt; End((B)) </pre>

#### 4.10.9. İkinci Dereceden Denklemlerin Köklerini Bulma

Algoritma	Akış Şeması
$ax^2 + bx + c = 0$ $\Delta = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ <pre> delta=b^2-4ac eğer delta&gt;0 ise     Evet,     x1=(-b+Vdelta)/2a     x2=(-b-Vdelta)/2a     yaz ,x1,x2     Hayır,     Eğer delta=0         Evet         x1=-b/2a         yaz, x1         Hayır,         Yaz, gerçek kök yok </pre>	 <pre> graph TD     Start([delta=b^2-4ac]) --&gt; DeltaGT0{delta&gt;0}     DeltaGT0 -- Evet --&gt; CalcX1X2[x1=(-b+Vdelta)/2a x2=(-b-Vdelta)/2a]     CalcX1X2 --&gt; PrintX1X2[yaz, x1,x2]     DeltaGT0 -- Hayır --&gt; DeltaEq0{delta=0}     DeltaEq0 -- Evet --&gt; CalcX1[x1=-b/2a]     CalcX1 --&gt; PrintX1[yaz, x1]     DeltaEq0 -- Hayır --&gt; End([Gerçek Kök Yok]) </pre>

#### 4.10.10. Mükemmel Sayı Kontrolü

Algoritma	Akış Şeması
<p>Bir sayının bölenlerinin toplamı kendine eşitse bu sayı mükemmel bir sayıdır. Örnek olarak 6 mükemmel bir sayıdır (<math>1 + 2 + 3 = 6</math>). 6,28,496,8128</p>	

#### 5. Çözümün Programlanması/Kodlanması

Akış şeması ve algoritmalar tamamlandıktan sonra istenilen bir programlama dili kullanılarak programın yazılması işlemine geçilir. Bu işleme “programlama” ya da “kodlama” adı verilir. Kodlama sonucunda programın ne kadar hatasız çalıştığı, algoritmanın etkililiğine bağlıdır.



## KAYNAKÇA

- **2007.** *personel.klu.edu.tr*. [Çevrimiçi] 17 10 2007.  
[http://personel.klu.edu.tr/dosyalar/kullanici/m.aslanyurek/dosyalar/dosya\\_ve\\_belgeler/Programlamaya%20Giri%C5%9F%28Algoritma%29%281%29.pdf](http://personel.klu.edu.tr/dosyalar/kullanici/m.aslanyurek/dosyalar/dosya_ve_belgeler/Programlamaya%20Giri%C5%9F%28Algoritma%29%281%29.pdf).
- **AYTEN, Umut Engin. 2010.** *http://www.yildiz.edu.tr*. [Çevrimiçi] 20 10 2010.  
[http://www.yildiz.edu.tr/~ayten/algortimaveprogramlama\\_bolum1-2.pdf](http://www.yildiz.edu.tr/~ayten/algortimaveprogramlama_bolum1-2.pdf).
- **BAŞER, Mustafa. 2017.** *Python*. İstanbul : Dikeyksen, 2017. 978-605-87588-7-2.
- **ÇAMOĞLU, Kadir. 2010.** *10 Adımda Yazılım Geliştirme*. basım yeri bilinmiyor : Kodlab, 2010.
- **2017.** [http://www.robotiksistem.com/programlama\\_dilleri\\_ozellikleri.html](http://www.robotiksistem.com/programlama_dilleri_ozellikleri.html). [Çevrimiçi] 17 09 2017.  
[http://www.robotiksistem.com/programlama\\_dilleri\\_ozellikleri.html](http://www.robotiksistem.com/programlama_dilleri_ozellikleri.html).
- **2017.** <https://tr.wikibooks.org/>. [Çevrimiçi] 05 01 2017.  
[https://tr.wikibooks.org/wiki/Programlama\\_Temelleri/Programlama\\_Ara%C3%A7lar%C4%B1](https://tr.wikibooks.org/wiki/Programlama_Temelleri/Programlama_Ara%C3%A7lar%C4%B1).
- **2017.** [https://tr.wikibooks.org/wiki/Programlama\\_Temelleri/Programlama\\_Ara%C3%A7lar%C4%B1](https://tr.wikibooks.org/wiki/Programlama_Temelleri/Programlama_Ara%C3%A7lar%C4%B1). [Çevrimiçi] 05 01 2017.  
[https://tr.wikibooks.org/wiki/Programlama\\_Temelleri/Programlama\\_Ara%C3%A7lar%C4%B1](https://tr.wikibooks.org/wiki/Programlama_Temelleri/Programlama_Ara%C3%A7lar%C4%B1).
- **2017.** [https://tr.wikipedia.org/wiki/Ada\\_Lovelace](https://tr.wikipedia.org/wiki/Ada_Lovelace). [Çevrimiçi] 19 Nisan 2017.  
[https://tr.wikipedia.org/wiki/Ada\\_Lovelace](https://tr.wikipedia.org/wiki/Ada_Lovelace).
- **KIZILÖREN, Tevfik. 2011.** *Java ve Java Teknolojileri*. İstanbul : Kodlab, 2011. ISBN 978-605-4205-25-7.
- **2017.** *Ortaöğretim Bilgisayar Bilimi Ders Kitabı Kur 1*. basım yeri bilinmiyor : MEB, 2017. s. 17.
- **2017.** [rapidtables.com](http://www.rapidtables.com/code/text/ascii-table.htm#print). [Çevrimiçi] 07 10 2017. <http://www.rapidtables.com/code/text/ascii-table.htm#print>.
- **TUNGUT, H. Burak. 2017.** *Algoritma ve Programlama Mantığı*. basım yeri bilinmiyor : Kodlab, 2017.