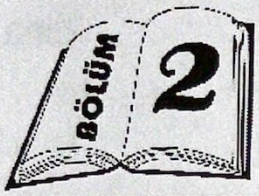
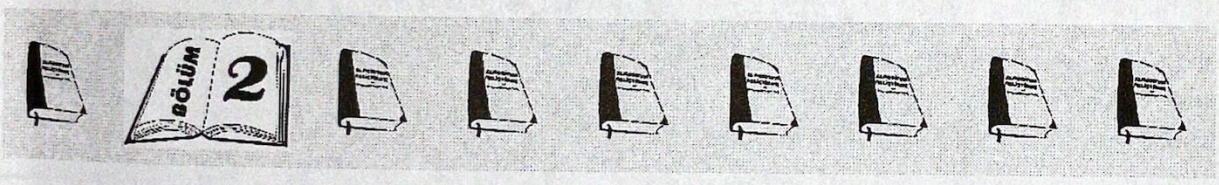


Bölüm 2

ALGORİTMA



Bu bölümde

Algoritma hakkındaki temel bilgileri,

Algoritmalarda kullanılan operatörleri ve terimleri,

Algoritma hazırlama aşamaları hakkındaki temel adımları öğreneceksiniz.

Bu bölüm sonunda

Algoritma kavramını açıklayabilecek,

Algoritmalardaki operatör ve terimleri açıklayabilecek ve kullanabilecek,

Algoritma hazırlama aşamalarını kavrayacak,

Verilen algoritmaları analiz edebilecek,

Verilen algoritmaların çalıştırma sonuçlarını elde edebileceksiniz.

I. GİRİŞ

İnsan, günlük yaşamında herhangi bir işi gerçekleştirmeden önce genellikle plan yapar. Plan; işin nasıl yapılacağını, hangi aşamalardan geçileceğini, ne tür adımlar izleneceğini düzenli bir şekilde belirtir ve hedefe ulaşmada hem yol gösterir hem de büyük kolaylıklar sağlar. Planlı bir şekilde gerçekleştirilen işlerde, başarıya ulaşmak çok daha kolaydır.

Bilgisayar dünyasında, insanın yaşamı boyunca yaptığı "*plan*" kavramına eşdeğer "*algoritma*" kavramı vardır. *Bilgisayardaki işlemin/işlemlerin gerçekleştirilmesinde izlenecek adımlara (adımlar dizisine)*, "**algoritma**" *denir*. Yani algoritma, işlemleri yaptırabilmek (problemleri çözdürebilmek, kontrolleri gerçekleştirebilmek vb.) için bilgisayara öğretilen/iletilen işlem basamaklarıdır. Bilgisayarın; bir problemi çözerken hangi ve neredeki giriş değerlerini alacağı, bunları işlerken ne tür yöntemleri kullanacağı, ne tür sonuçlar üreteceği ve bu sonuçları nerede göstereceği veya saklayacağı vb. adımların hepsi, hazırlanan algoritmanın herhangi bir programlama dilinin kurallarına uygun olarak yazılmış komutlarıyla (programlarla) bilgisayara iletilir. *Algoritmanın özel geometrik şekillerle çizilmiş hali de* "**akış diyagramı**" *olarak adlandırılır*.

Bilgi

"Algoritma" kelimesi Orta Asya'daki Harezm'de doğan matematikçi *Ebu Abdullah Muhammed bin Musa el Harezmi*'nin isminden gelmektedir. IX. yüzyılda cebir alanında yayınladığı "Hisab el-cebir ve el-mukabala" kitabı, dünyadaki cebir alanında ilk kitaplardandır. Latince'ye tercüme edilmesiyle çok büyük ilgi görür ve Avrupalılar "el-Harezmi" (Arapça'da Al-Khwârizmî) yerine "*algorizm*" kelimesini 'Arap rakamlarını kullanarak aritmetiksel problemleri çözme kuralları' anlamında kullanırlar ve zamanla bu kelime "*algoritma*"ya dönüşür.

Program yazılırken aşağıdaki adımlara dikkat edilmesi, programcının yararına olacaktır (Şekil 2.1).

Program yazma adımları

i. Problem/iş iyice irdelenir/incelenir (analiz edilir).

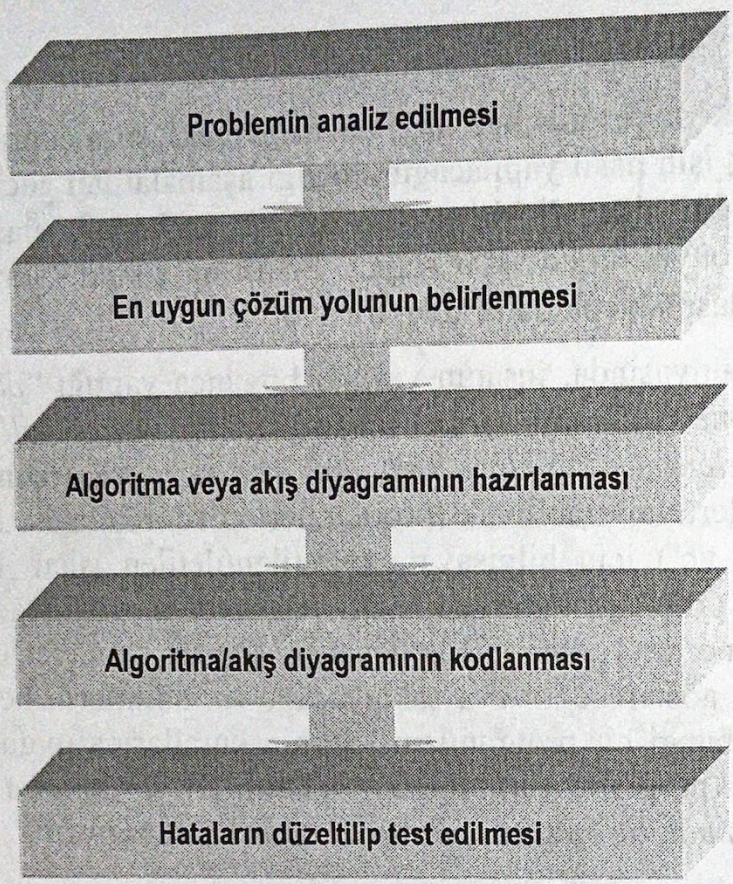
ii. Çözüm yolları ortaya konularak programlamaya en uygun (en az komutla, en kısa sürede, en doğru-hassas sonuç veren) çözüm yolu seçilir/belirlenir.

iii. Programın algoritması hazırlanır veya akış diyagramı çizilir.

iv. Algoritma veya akış diyagramı, programcının bildiği veya tercih ettiği bir programlama diliyle kodlanır.

v. Program, kodlandığı editörde çalıştırılarak yazım hataları düzeltilir.

vi. Bilinen giriş-çıkış değerleriyle programın doğru çalışıp çalışmadığı test edilir (doğrulama yapılır).



Şekil 2.1: Program yazma adımları

Algoritmanın hazırlanması veya akış diyagramının çizilmesi, program yazma işleminin temelidir. Hangi programlama dilinde program yazılıyor olsa da algoritma/akış diyagramı değişmemektedir. Bu nedenle algoritması hazırlanmış/yazılmış veya akış diyagramı çizilmiş bir problem, tüm programlama dillerinde kodlanarak programı kolaylıkla yazılabilir.

Algoritmalar genel olarak üç şekilde ifade edilebilirler (gösterilebilirler-sunulabilirler):

a. *Algoritmanın metin olarak yazılması*: Problemin çözüm adımları, düz metin olarak açık cümlelerle ifade edilebilir.

b. *Algoritmanın sözde kodlarla (pseudocode) yazılması*: Problemin çözüm adımları, komut benzeri anlaşılır metinlerle ifade edilebilir.

c. *Algoritmanın akış diyagramının çizilmesi*: Problemin çözüm adımları, geometrik şekillerle gösterilebilir.

II. ALGORİTMALARDA KULLANILAN OPERATÖRLER

İşlemleri belirten simgelere, bilgisayar terminolojisinde "*operatör*" denir. Algoritmalarda kullanılan operatörler, Tablo 2.1'de toplu olarak verilmektedir.

Tablo 2.1: Algoritmalarda kullanılan operatörler

OPERATÖRLER	
Matematiksel işlem operatörleri	
^	Üs alma
*	Çarpma
/	Bölme
+	Toplama
-	Çıkarma
.	Tam ve ondalıklı kısımları ayırma
Karşılaştırma operatörleri	
=	Eşittir
<>	Eşit değildir
<	Küçüktür
>	Büyüktür
>=	Büyük eşittir
=>	Büyük eşittir
<=	Küçük eşittir
=<	Küçük eşittir
Mantıksal işlem operatörleri	
`	DEĞİL
.	VE
+	VEYA
Küme işlem operatörleri	
+	Birleşim
-	Fark
*	Kesişim
=	Eşit
<>	Eşit değil
=<	Soldaki küme, sağdakinin alt kümesi
<=	Soldaki küme, sağdakinin alt kümesi
>=	Sağdaki küme, soldakinin alt kümesi
=>	Sağdaki küme, soldakinin alt kümesi
Alfasayısal operatörler	
+	Birleştirme
Genel işlem operatörleri	
=	Aktarma
( )	Parantez

Not

Her programlama dilinde bu operatörlerin yazımında farklılıklar olabileceği gibi, ilave operatörler de bulunabilir.

III. ALGORİTMALARDA KULLANILAN TERİMLER

Algoritmalarda (programlarda) sık sık '*tanımlayıcı*', '*değişken*', '*sabit*', '*aktarma*', '*sayaç*', '*döngü*' vb. terimler kullanılır. Bundan sonraki kısımda ilgili terimler, sırasıyla açıklanacaktır.

A. Tanımlayıcı

Programı yazan kişi tarafından düşünülüp oluşturulan ve programdaki değişkenleri, sabitleri, paragrafları, kayıt alanlarını, özel veri/bilgi tiplerini, alt programları vb. adlandırmak için kullanılan kelimelere/ifadelere "*tanımlayıcı*" denir. Prensip olarak; programdaki tanımlayıcı kelimelerin, yerini tuttukları ifadelere çağrışım yapacak şekilde seçilmesi/oluşturulması daha uygundur. Böylece diğer programcıların kodları incelemesi, anlaması ve geliştirmesi/düzenlemesi daha da kolaylaşacaktır. Bu kelimeler oluşturulurken, aşağıdaki "*isimlendirme kuralları*"na uyulmalıdır.

İsimlendirme kuralları

i. İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir.
ii. 0-9 arası rakamlar kullanılabilir.
iii. Simgelerden sadece alt çizgi(_) kullanılabilir.
iv. Tanımlayıcı isimleri, harf veya alt çizgi ile başlayabilir.
v. Tanımlayıcı ismi, rakamla başlamamalı veya sadece rakamlardan oluşmamalıdır (sayı olmamalıdır).
vi. Tanımlayıcı ismi, ilgili programlama dilinin komutu veya saklı kelimelerinden olmamalıdır.

B. Değişken

Programın her çalıştırılmasında, farklı değerler alabilen/aktarılabilen bellek/veri/bilgi alanları, "*değişken*" olarak adlandırılır.

"İsimlendirme kurallarına" uyularak değişken isimleri oluşturulur. Bu tamamen programcının isteğine bağlıdır. Ancak değişken ismin; yerini aldığı ifadeye çağrışım yapacak şekilde olması, programın anlaşılırlığı açısından önemlidir.

Örneğin; bir kişinin adının aktarıldığı değişken '`Ad`', adı ve soyadının aktarıldığı '`Adsoy`', telefonunun aktarıldığı '`tel`', ev adresinin aktarıldığı '`EvAdres`', işyeri adresinin aktarıldığı '`isadres`', vb. şeklinde oluşturulabilir.

Örnek-2.1:

```
1. Başla
2. Birinci sayıyı (A) gir
3. İkinci sayıyı (B) gir
4. C=A+B işlemini yap
5. Sonucu (C) yaz
6. Dur
```

Yukarıdaki algoritma, dışarıdan (klavyeden) girilen iki sayının toplamını yapıp sonucu ekrana yazdırır. Algoritmada '`A`', '`B`' ve '`C`' değişkendir. Program her çalıştırıldığında '`A`' ve '`B`' değişkenlerine farklı değerler girilir ve sonucu temsil eden '`C`' de farklı değerler oluşur (Tablo 2.2).

Tablo 2.2: Örnek-2.1 için uygulama sonuçları

Girilen A sayısı	Girilen B sayısı	C=A+B	Yazılan C sayısı	
-3	3	0	0	→ 1. çalıştırma
3	11	14	14	→ 2. çalıştırma
33	99	132	132	→ 3. çalıştırma
1	2	3	3	→ 4. çalıştırma
-100	200	100	100	→ 5. çalıştırma
50	-83	-33	-33	→ 6. çalıştırma

Örnek-2.2:

```
1. Başla
2. Öğrencinin numarasını (No) gir
3. Öğrencinin adını ve soyadını (Adsoy) gir
4. Öğrencinin vize notunu (Vize) gir
5. Öğrencinin final notunu (Final) gir
6. Ort = 0.3*Vize + 0.7*Final
7. Numara (No) ve ortalamayı (Ort) yaz
8. Dur
```

Verilen algoritmada; öğrencinin numarası, adı ve soyadı, vize notu ve final notu girilmektedir. Verilere göre vize notunun %30'ü ile final notunun %70'i alınarak ortalaması hesaplanmakta ve ekrana yazdırılmaktadır. Burada değişken olarak `No`, `Adsoy`, `Vize`, `Final` ve `Ort` vardır. Programın çalıştırılmasıyla oluşabilecek bazı sonuçlar Tablo 2.3'te verilmektedir.

Tablo 2.3: Örnek-2.2 için uygulama sonuçları

No	Adsoy	Vize	Final	Ort	Ekrana yazılan	
9801.0033	Ahmet	30	70	58	9801.0033 – 58	→ 1. çalıştırma
9804.0099	Mehmet	70	90	84	9804.0099 – 84	→ 2. çalıştırma
9712.0123	Ali	60	30	39	9712.0123 – 39	→ 3. çalıştırma

C. Sabit

Birçok alanın kendi ait sabitleri bulunmaktadır. Örneğin matematikte π sayısı, doğal logaritma tabanı olan e sayısı bunlara ilişkin bazı örneklerdir. Programlardaki değeri değişmeyen ifadelere "*sabit*" denir. "İsimlendirme kuralları"na uygun olarak oluşturulan sabit isimlerine program/algoritma içinde sayısal veriler doğrudan; alfasayısal veriler ise tek/çift tırnak içinde aktarılır.

Örnek-2.3:

```
1. Başla
2. pi_sayisi=3.14
3. Dairenin yarıçapını (r) gir
4. cevre=2*pi_sayisi*r
5. alan=pi_sayisi*r*r
6. Çevre(cevre) ve alanı(alan) yaz
7. Dur
```

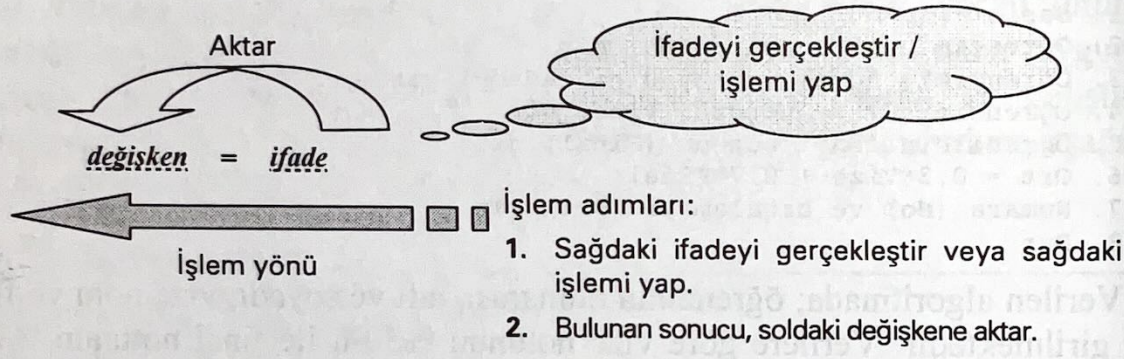
Verilen algoritmada; dairenin yarıçapı girilmekte ve çevresiyle alanı hesaplatılıp yazdırılmaktadır.

D. Atama/aktarma

Herhangi bir bilgi alanına, veri yazma; herhangi bir ifadenin sonucunu başka bir değişkende gösterme vb. görevlerde "*atama*" veya "*aktarma*" operatörü kullanılır.

```
değişken = ifade
```

satırında '`değişken`' yazan kısım, herhangi bir değişkenin/sabitin ismidir. '`ifade`' yazan kısımda ise; matematiksel, mantıksal veya alfasayısal ifade olabilir. Aradaki ' = ' sembolü, "*atama operatörü*" veya "*aktarma operatörü*" olarak adlandırılır ve sağdaki ifadenin/işlemin sonucu soldaki değişkene aktarır. Bu durumda değişkenin - eğer varsa - bir önceki değeri (eski değeri) silinir (Şekil 2.2).



Şekil 2.2: Aktarma işlemi

Not

Atama operatörü, eşitlik operatörü ile karıştırılmamalıdır. "=" atama operatörü, "←" anlamındadır.

Aktarma işlemiyle ilgili örnekler Tablo 2.4'te görülmektedir. Bu örneklerde her ifade 3 kez arka arkaya çalıştırılmaktadır. Bulunan her yeni sonuçta, eski sonucun hiçbir etkisi yoktur. Çünkü bulunan her yeni sonuç; eskisinin üzerine yazılmakta, yani onu silmektedir. Yani "`c=a+b`" ifadesi "`c←a+b`" anlamındadır.

Tablo 2.4: Aktarma işlemi uygulama sonuçları

a	b	Eski c	Yeni c	
Matematiksel ifadelerde (*c = a + b*)				
3	7	---	*10*	→ 1. çalıştırma
5	7	*10*	*12*	→ 2. çalıştırma
20	33	*12*	*53*	→ 3. çalıştırma
Mantıksal ifadelerde (*c = a + b*)				
0	1	---	*1*	→ 1. çalıştırma
0	0	*1*	*0*	→ 2. çalıştırma
1	1	*0*	*1*	→ 3. çalıştırma
Alfasayısal ifadelerde (*c = a + b*)				
Galata	saray	---	Galatasaray	→ 1. çalıştırma
Beşik	taş	*Galatasaray*	*Beşiktaş*	→ 2. çalıştırma
Fener	bahçe	*Beşiktaş*	*Fenerbahçe*	→ 3. çalıştırma

Tablo 2.4'te, matematiksel mantığa aykırı bir durum yoktur.

```
TPL = TPL + 1
```

işlemine, matematiksel mantıkla yaklaşılırsa;

```
TPL = TPL+1
TPL - TPL = 1
0 = 1
```

şeklinde yanlış bir sonuç çıkar. Bilgisayar dilinde durum, böyle değildir. Önce sağ taraftaki ifadede değişkenin eski/mevcut değeriyle işlem gerçekleştirilmekte; daha sonra bulunan sonuç, sol tarafta yeni değer olarak aktarılmaktadır (eskisinin üzerine yazılmaktadır). Yani aradaki "="; matematiksel eşittir operatörü değil, atama/aktarma operatörüdür.

Örnek 2.4:

```
1. Başla
2. T=0
3. Bir sayı (A) gir
4. T=T+A işlemini yap
5. Başka bir sayı (B) gir
6. T=T+B işlemini yap
7. Başka bir sayı (C) gir
8. T=T+C işlemini yap
9. Yaz T
10. Dur
```

Yandaki algoritmada; klavyeden üç sayı girilip toplanmaktadır. Toplama işlemi `T=A+B+C` ile de yapılabilir. Burada, aktarma işleminin açıklanması açısından bu ifade kullanılmamıştır. Algoritmada her sayı girildikten sonra `T` değişkeninde toplanmakta ve program sonunda değeri, ekrana yazdırılmaktadır. Bu örneğin bir çalıştırma sonucu ve işlem adımları Tablo 2.5'te verilmektedir.

Tablo 2.5: Örnek-2.4 için çalıştırma sonuçları

İşlem sırası	A	B	C	Eski T	Yeni T
1	3	---	---	0	0+3=3
2	---	4	---	3	3+4=7
3	---	---	5	7	7+5=12
					T=12

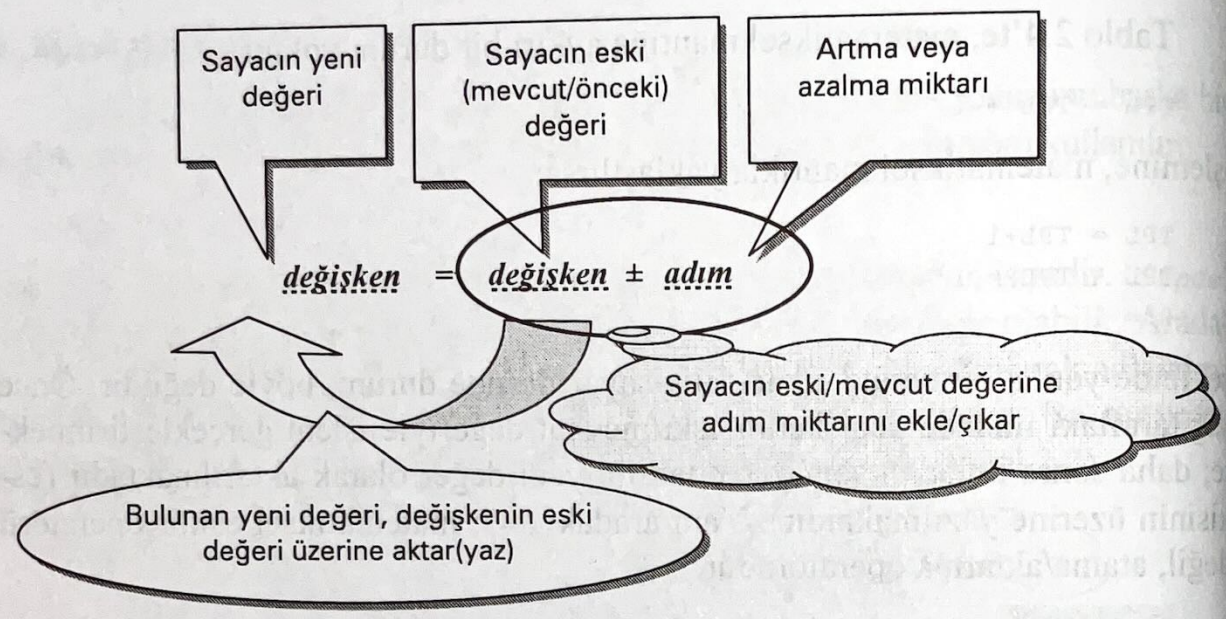
E. Sayaç

Programlarda; bazı işlemlerin belirli sayıda yapılması veya işlenen/üretilen değerlerin sayılması gerekebilir. Örneğin klavyeden girilen bir cümlede kaç sesli harf olduğunu bulan programda, cümlenin her harfi sırayla çağırılır ve sesli harfler kümesine ait olup olmadığı araştırılır. Eğer sıradaki çağırılan harf bu kümeye ait ise bunları sayacak olan değişkenin değeri bir arttırılır.

```
Say = Say + 1
```

şeklindeki işlemde sağdaki ifadede değişkenin eski (mevcut/önceki) değerine '1' eklenmekte; bulunan sonuç yine kendisine, yeni değer olarak aktarılmaktadır. Bu tür sayma işlemlerine, algoritmada "*sayaç*" veya "*sayıcı*" adı verilir. Yani "*sayaç*"; işlem akışı kendisine her geldiğinde, belirtilen adım değeri kadar artan/azalan değişkendir. Algoritmalarda sayacın genel kullanımı Şekil 2.3'teki gibidir.

```
sayaç değişkeni = sayaç değişkeni ± adım
```



Şekil 2.3: Sayaç işlemi

Örneğin `X=X+3` üçer üçer artan, `S=S-5` beşer beşer azalan bir sayıcıdır.

Örnek-2.5: Aşağıdaki algoritmada 1-5 arası sayılar, ekrana yazdırılmaktadır. 1-5 arası sayıları oluşturmak için sayaç (`S=S+1`) kullanılmakta ve algoritmanın çalışma prensibi Tablo 2.6'da verilmektedir.

```
1. Başla
2. S=0
3. Eğer S>4 ise git 7
4. S=S+1
5. Yaz S
6. Git 3
7. Dur
```

Tablo 2.6 Örnek-2.5'teki sayacın çalışması

Eski S	Yeni S	Ekrana yazılan
0	0+1=1	1
1	1+1=2	2
2	2+1=3	3
3	3+1=4	4
4	4+1=5	5

F. Döngü

Birçok programda; bazı işlemler, belirli ardışık değerlerle gerçekleştirilmekte veya belirli sayıda yapılmaktadır. Programlardaki belirli işlem bloklarını, verilen sayıda gerçekleştiren işlem akış çevrimlerine "*döngü*" denir. Örneğin 1 ile 1000 arasındaki tek sayıların toplamını hesaplayan programda `T=1+3+5+7+9+11+...` gibi uzun uzun yazmak yerine 1 ile 1000 arasında ikişer ikişer artan bir döngü açılır ve döngü değişkeni ardışık olarak toplanır.

Döngü oluşturma kuralları

i. Döngü değişkenine başlangıç değeri verilir.

ii. Döngünün artma veya azalma miktarı belirlenir.

iii. Döngünün bitiş değeri belirlenir.

iv. Eğer döngü, karar/karşılaştırma ifadeleriyle oluşturuluyorsa; döngü değişkeni, döngü içinde adım miktarı kadar arttırılmalı/azaltılmalıdır.

Örnek-2.6: Aşağıdaki algoritma 1-10 arası tek sayıların toplamını hesaplamaktadır ve çalışma prensibi Tablo 2.7'de verilmektedir.

Tablo 2.7 Örnek-2.6'daki döngünün çalışması

Eski J	Eski T	Yeni T	Yeni J
1	0	0+1=1	3
3	1	1+3=4	5
5	4	4+5=9	7
7	9	9+7=16	9
9	16	16+9=25	11
11	-	-	-

```
1. Başla
2. T=0
3. J=1
4. Eğer J>10 ise git 8
5. T=T+J
6. J=J+2
7. Git 4                    Döngü
8. Yaz T
9. Dur
```

G. Ardışık Toplama

Ardışık (ardışıl) toplama işleminin çalışma prensibi, sayacınkine benzemektedir. Programlarda, aynı değerin üzerine yeni değerler eklemek için kullanılır. Genel kullanım şekli aşağıdaki gibidir:

```
Toplam değişkeni = Toplam değişkeni + Sayı
```

Not

Toplam değişkenine, başlangıç değeri olarak toplama işleminin etkisiz elemanı olan **0** (sıfır) atanır.

Örnek-2.7: Klavyeden girilen 5 sayının ortalamasını bulan programın algoritması.

N adet sayının ortalaması, bu sayıların toplamının N'e bölünmesi ile elde edilir. Aşağıdaki algoritmada `N=5` olup sayıların toplamı için '`T`' değişkeni; beş sayının girilip girilmediğini kontrol etmek için ise (saymak için) '`S`' sayacı kullanılmış ve çalışma prensibi Tablo 2.8'de verilmiştir.

```
1. Başla
2. T=0
3. S=0
4. Eğer S>4 ise git 9
5. S=S+1
6. Sayıyı (A) gir
7. T = T + A
8. Git 4
9. Ortalama=T/N
10. Yaz Ortalama
11. Dur
```

Tablo 2.8 Örnek-2.7 uygulama sonuçları

Eski S	Yeni S	Sayı	Eski T	Yeni T	Ort.
0	0+1=1	3	0	0+3=3	---
1	1+1=2	1	3	3+1=4	---
2	2+1=3	5	4	4+5=9	---
3	3+1=4	11	9	9+11=20	---
4	4+1=5	35	20	20+35=55	---
Ortalama=55/5=11					

H. Ardışık Çarpma

Ardışık çarpma işleminde; aynı değer, yeni değerlerle çarpılarak eskisinin üzerine aktarılmaktadır. Genel kullanım şekli aşağıdaki gibidir:

```
Çarpım değişkeni = Çarpım değişkeni * Sayı
```

Not

Çarpım değişkenine, başlangıç değeri olarak çarpma işleminin etkisiz elemanı olan **1** (bir) atanır.

Ardışık toplama veya çarpma işleminde başlangıç değeri olarak, yapılan işlemin etkisiz elemanı verilir. Çarpma işleminde; çarpım değişkenine başlangıç değeri olarak '0' verilirse; bir sayının sıfır ile çarpımı sıfır olduğu için sonuç, daima sıfır olur.

Örnek-2.8: Klavyeden girilen N sayısının faktöriyelini hesaplayan programın algoritması.

Aşağıdaki algoritmada, klavyeden bir N sayısı girilmekte ve bunun faktöriyeli hesaplanmaktadır. N=5 için programın çalışması Tablo 2.9'da verilmektedir.

```
1. Başla
2. N sayısını gir
3. Fak=1
4. S=0
5. Eğer S>N-1 ise git 9
6. S=S+1
7. Fak=Fak*S
8. Git 5
9. Yaz Fak
10. Dur
```

Tablo 2.9: Örnek-2.8 için uygulama sonuçları

Eski S	Yeni S	Eski Fak	Yeni Fak	Sonuç
0	0+1=1	1	1*1=1	---
1	1+1=2	1	1*2=2	---
2	2+1=3	2	2*3=6	---
3	3+1=4	6	6*4=24	---
4	4+1=5	24	24*5=120	120

IV. ALGORİTMA HAZIRLAMA

Daha önce de değinildiği gibi, "*algoritma, programın temelidir*". Algoritması hazırlanmış (veya akış diyagramı çizilmiş) bir programın kodunu yazmak çok kolaydır. İlgili programlama dilinin kaynaklarına bakılarak kodlamalar rahatlıkla gerçekleştirilebilir. Algoritma hazırlarken aşağıda verilen adımlar izlenebilir.

Algoritma hazırlama kuralları

i. Yapılacak iş/çözülecek problem iyice irdelenir. Tüm olasılıklar gözden geçirilir.

ii. En az komutla, en kısa sürede, en doğru-hassas sonuca ulaştıracak çözüm yolu/yöntem veya sadece çözüm yolu/yöntem belirlenir.

iii. Tanımlayıcı isimleri belirlenir.

iv. Algoritmada her işlem adımına bir numara verilir.

v. Problemin çözümü için gerekli olan veriler/işlenecek veriler girilir veya başka ortamlardan alınır.

vi. Yapılacak işlemler/kullanılacak yöntemler açık şekilde verilir.

vii. Bulunan sonuçlar görüntülenir veya belirli ortamlarda saklanır.

Not

Algoritma, bilgisayarın gerçekleştireceği işlem adımlarını açık ve tam olarak anlatmalıdır.

Algoritma hazırlama kuralları (özet)

i. Tanımlayıcı isimleri belirlenir.

ii. Veri girişleri yapılır veya çevre birimlerden okutulur.

iii. Yapılacak işlemler ve yöntemler yazılır.

iv. Sonuçlar yazdırılır veya çevre birimlerde saklanır.

Algoritmanın/akış diyagramının avantajları

i. Program yazmayı kolaylaştırır.

ii. Hatalı kodlama oranını azaltır.

iii. Program yazımı için geçen süreyi kısaltır.

iv. İşlem akışını açık bir şekilde gösterdiğinden program kontrolünü ve hata takibini kolaylaştırır.

v. Sonradan yapılacak düzenlemelerde kolaylıklar sağlar.

Not

Algoritma;

i. Tüm işlem adımlarını ve ayrıntıları, sıralı bir şekilde anlatmalıdır.

ii. Esnek olmalıdır.

iii. Değişikliklere/güncellemelere/geliştirmelere uygun olmalıdır.

iv. Programlamaya uygun olup herhangi bir programlama diline bağımlı olmamalıdır.

Not

İyi bir program;

i. İşlemleri, hatasız gerçekleştirip doğru sonuçlar üretmelidir.

ii. Hızlı olmalıdır.

iii. Türüne göre bellekte fazla yer kaplamamalıdır.

iv. Sistem kaynaklarını gereksiz kullanmamalıdır.

v. Kodları sade olup değişiklikler/güncelleştirmeler/geliştirmeler kolaylıkla yapılabilmelidir.

vi. Modüler olup diğer programlarla da uyumlu bir şekilde çalışabilmelidir.

SORULAR

1. Algoritma nedir? Algoritmanın gerekliliğini ve avantajlarını açıklayınız.

2. Program yazılmadan önce algoritması mı hazırlanmalı yoksa akış diyagramı mı çizilmelidir?

3. Algoritma hazırlanırken dikkat edilmesi gereken hususları açıklayınız.

4. Değişken nedir? Programlarda neden değişkenlere ihtiyaç duyulmaktadır?

5. Sayaçlar, nerelerde ve niçin kullanılmaktadır?

6. Aşağıdaki algoritmanın sonucu nedir?

```
1. Başla
2. T=0
3. S=0
4. Eğer S>10 ise Git 8
5. T=T+2*S
6. S=S+2
7. Git 4
8. Yaz T
9. Dur
```

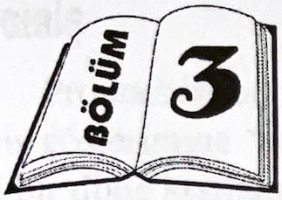
7. Aşağıdaki algoritmanın sonucunu hesaplayınız?

```
1. Başla
2. F=1
3. S=20
4. Eğer S<1 ise Git 9
5. S= S-3
6. F= F+S
7. F= F+2
8. Git 4
9. Yaz F
10. Dur
```

8. Girilen üç sayıdan en büyüğünü bulan programın algoritmasını hazırlayınız.

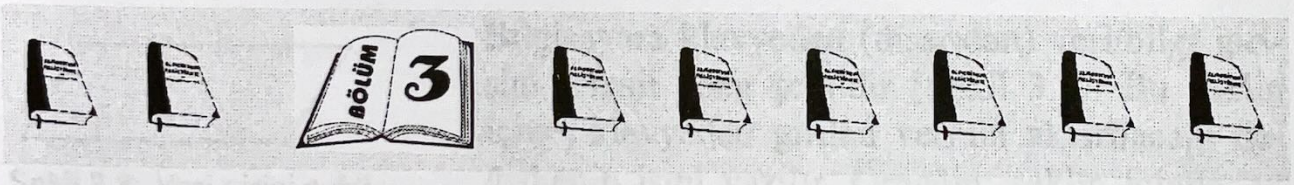
9. Girilen üç sayıyı küçükten büyüğe doğru sıralayan programın algoritmasını hazırlayınız.

10. 1-99 arasındaki tek ve çift sayıların toplamları ile çarpımlarını ayrı ayrı hesaplayan programın algoritmasını hazırlayınız.



Bölüm 3

AKIŞ DİYAGRAMI



Bu bölümde

Akış diyagramlarında kullanılan temel şekilleri,

Akış diyagramlarını analiz edip üreteceği sonuçları elde etmeyi,

Akış diyagramlarının çizilmesindeki temel adımları öğreneceksiniz.

Bu bölüm sonunda

Akış diyagramlarındaki şekilleri açıklayabilecek,

Verilen akış diyagramlarını analiz edebilecek,

Verilen akış diyagramlarının çalıştırma sonuçlarını elde edebilecek,

Basit işlemlere ait akış diyagramlarını çizebileceksiniz.

I. GİRİŞ

Programlamanın temeli olan algoritmanın, özel geometrik şekillerle çizilmiş gösterimine '*akış diyagramı*' denilmektedir. Yazılımcı; programını herhangi bir dilde kodlamadan önce onun algoritmasını yazabilir veya akış diyagramını çizebilir. Akış diyagramı şekilleri, en genel gösterim biçimleriyle bu bölümde ayrıntılı olarak anlatılacak ve daha sonraki bölümlerde program kodlarıyla birlikte verilecektir.

II. AKIŞ DİYAGRAMINDA KULLANILAN ŞEKİLLER

Akış diyagramında kullanılan temel şekiller Tablo 3.1'de toplu olarak verilmektedir.

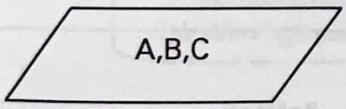
A. Başla/Dur

Şekil 3.1: Başla/Dur şekilleri

'`Başla`' ve '`Dur`' işlemleri, standarttır. Her akış diyagramı; '`Başla`' şekli ile başlar, '`Dur`' şekli ile biter (Şekil 3.1). İşlemler, bu iki şekil arasında açıklanır.

B. Veri Girişi

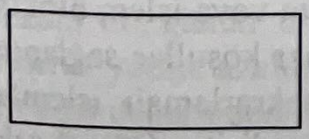
Şekil 3.2: Veri girişi şekli



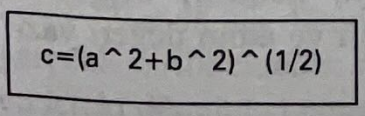
Şekil 3.3: Veri girişi örneği

Bilgisayara klavyeden (dışarıdan) veri/bilgi girişini temsil eden şekildir (Şekil 3.2). Bu şeklin içine, klavyeden girilen verinin aktarılacağı değişkenin ismi yazılır. Örneğin; şeklin içinde '`N`' yazıyorsa ve program çalıştırıldığında '`10`' girilirse, bu '`N=10`' olacak anlamındadır. Veri girişi şeklinin içine, birden fazla değişken ismi de yazılabilir. Şekil 3.3'teki giriş değişkenleri programlama dili ile kodlanıp çalıştırılırsa; işlem (program akışı) bu satıra geldiğinde klavyeden '`3`', '`11`' ve '`Teknik`' ifadeleri girilirse, yapılan işlem `A=3`, `B=11` ve `C='Teknik'` atamalarına eşdeğerdir.

C. İşlem



Şekil 3.4: İşlem şekli



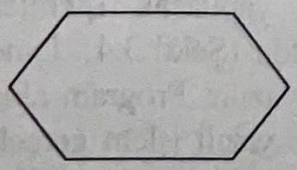
Şekil 3.5: İşlem örneği

Programın çalışması sırasında yapılacak işlemleri ifade etmek için kullanılan şekildir (Şekil 3.4). İçine, işlem cümleleri/ifadeleri aynen yazılır. Program akışı buraya geldiğinde, şeklin içinde yazılı işlem gerçekleştirilir. Birden fazla işlem; aynı şekil içinde, aralarına virgül konularak veya alt alta yazılarak gösterilebilir. Şekil 3.5'teki örnekte; işlem akışı bu şekle gelince, program $c=\sqrt{a^2+b^2}$ işlemini yapar.

Tablo 3.1: Akış diyagramlarında kullanılan genel şekiller

Şekil	Anlamı
	Başla / Dur
	Bilgi/veri girişi
	İşlem
	Döngü
	Karar/Karşılaştırma
	Bilgi/veri yazma
	Önceden tanımlı işlem (alt program, fonksiyon...)
	Bağlantı
	İşlem akış yönü

D. Döngü



Şekil 3.6: Döngü şekli

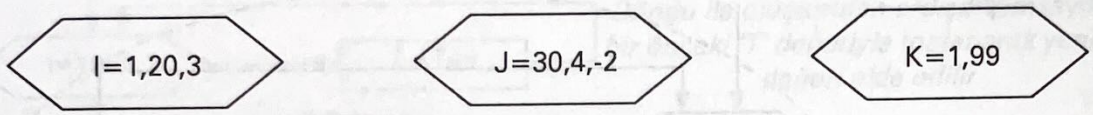
Birçok programda; belirli işlem veya işlem blokları ardışık değerlerle ya da bazı koşullar sağlanıncaya kadar tekrarlanır. Bu tekrarlamalı işlemler '*döngü*' (*çevrim*) olarak isimlendirilir. Döngü şeklinin içine; döngü (çevrim, kontrol) değişkeni, başlangıç değeri, bitiş değeri ve adım değeri yazılır (Şekil 3.6-3.7). Genel yazım şekli, aşağıdaki gibidir:

```
Döngü değişkeni = Başlangıç değeri, Bitiş değeri, Adım değeri
```

Döngüler, iki şekilde oluşturulabilir:

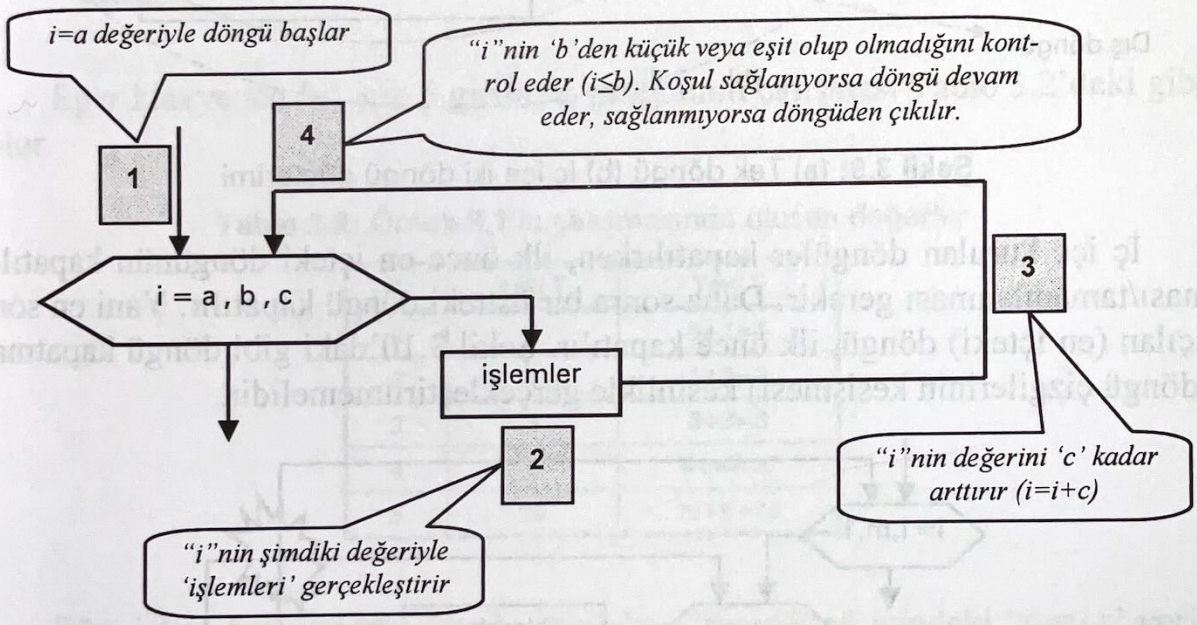
a. *Artan döngü*: Başlangıç değeri, bitiş değerinden küçüktür (adım değeri pozitiftir).

b. *Azalan döngü*: Başlangıç değeri, bitiş değerinden büyüktür (adım değeri negatiftir).



1'den 20'ye kadar 3'er artan 'I' döngüsü | *30'dan 4'e kadar 2'şer azalan 'J' döngüsü* | *1'den 99'a kadar 1'er artan 'K' döngüsü*

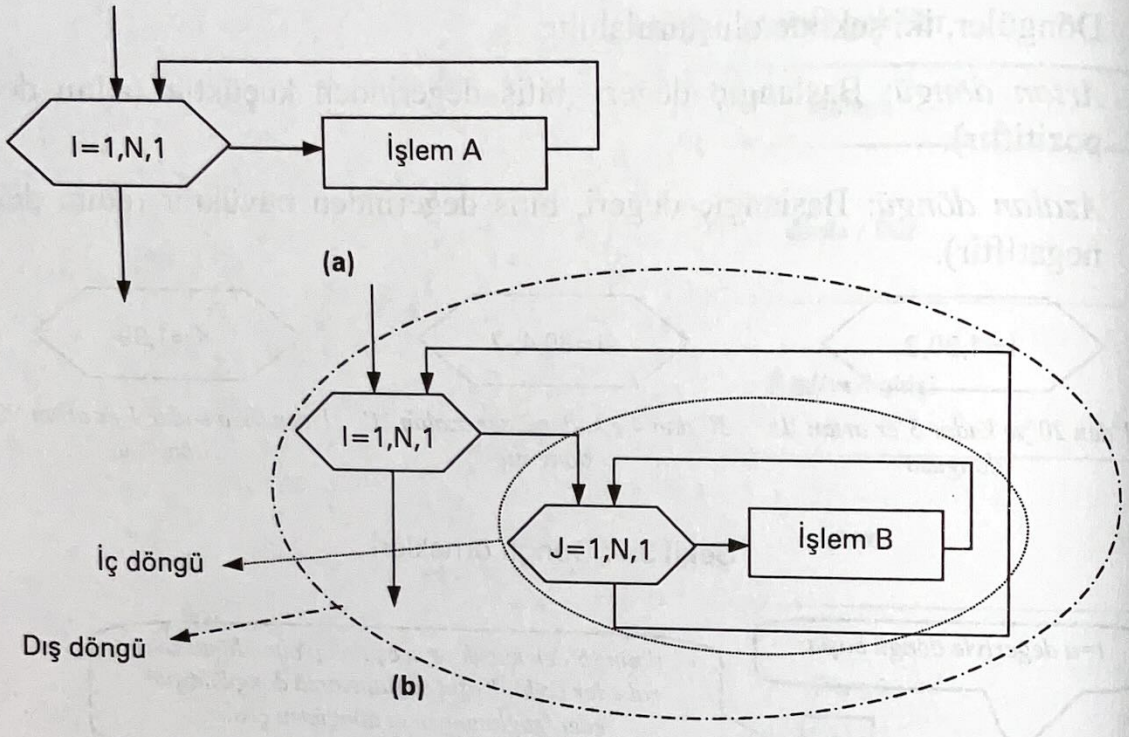
Şekil 3.7: Döngü örnekleri



Şekil 3.8: Örnek artan bir döngünün çalışma mantığı

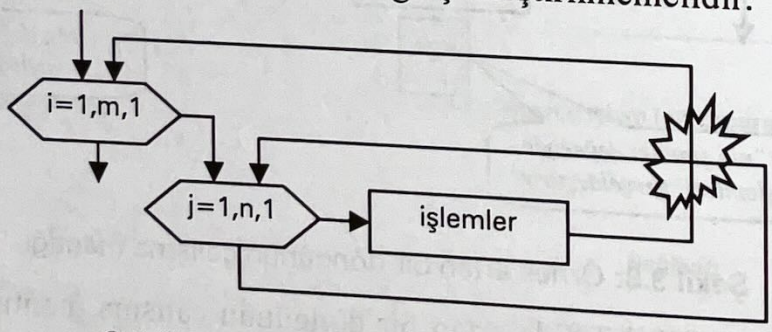
Örnek olarak Şekil 3.8'de artan bir döngünün çalışma mantığı verilmektedir. Döngü başladığında `i=a` başlangıç değeriyle döngü bloğu içindeki işlemler gerçekleşir. Sonraki aşamada döngü değişkeni, adım miktarı kadar otomatik olarak arttırılır (`i=i+c`). Son aşamada ise döngü değişkeninin, bitiş değerine ulaşıp ulaşmadığı(`i≤b`) yine otomatik olarak kontrol edilir. Eğer bitiş değeri, aşıldıysa döngü sonlanır (döngüden çıkılır); aşılmadıysa döngü devam eder. Azalan döngünün de çalışması benzer şekildedir; sadece döngü değişkenine arttırma yerine azaltma işlemi uygulanmakta, koşul olarak da döngü değişkeninin bitiş değerinden büyük veya eşit olup olmadığı kontrol edilmektedir.

Birçok programlarda iç içe birden fazla döngü kurulmaktadır. Örnek olarak Şekil 3.9.a'da tek döngü yapısı verilmekte ve bu döngüdeki '*İşlem A*', "`N`" defa gerçekleştirilmektedir. Şekil 3.9.b'de ise iç içe iki döngü bulunmaktadır. Önce en içteki döngü (`J`) tamamlanmakta ve '*İşlem B*', `N.N=N`^{2} defa gerçekleştirilmektedir.



Şekil 3.9: (a) Tek döngü (b) İç içe iki döngü gösterimi

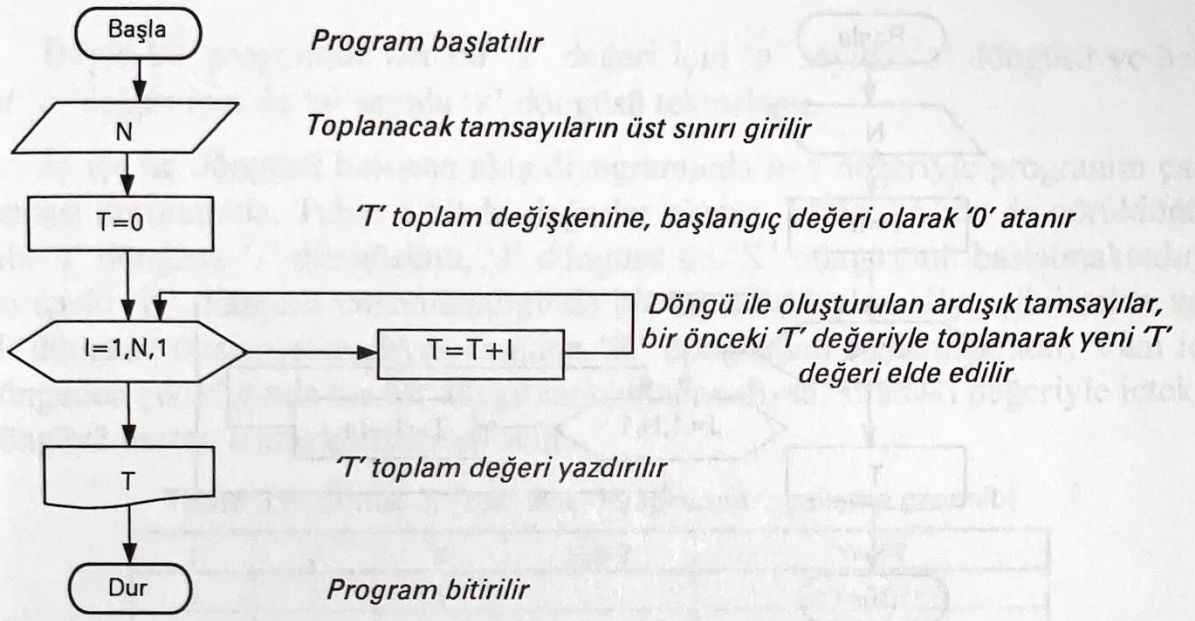
İç içe kurulan döngüler kapatılırken, ilk önce en içteki döngünün kapatılması/tamamlanması gerekir. Daha sonra bir üstteki döngü kapatılır. Yani en son açılan (en içteki) döngü, ilk önce kapatılır. Şekil 3.10'daki gibi döngü kapatma (döngü çizgilerinin kesişmesi) kesinlikle gerçekleştirilmemelidir.



Şekil 3.10: Döngülerin yanlış kapatılması

Örnek-3.1: 1'den N'e kadar olan tamsayıların toplamını hesaplayan programın akış diyagramı.

Öncelikle çizilecek olan akış diyagramında kullanılacak değişken isimleri belirlenir: toplanacak tamsayıların üst sınırı 'N', 1'den üst sınıra kadar olan tamsayıların toplamı 'T' ve sayıları üreten döngü değişkeni de 'I'. Birinci adımda toplanacak olan tamsayıların üst sınırının klavyeden girilmesi istenir. İkinci adımda ardışık toplama değişkenine sıfır değeri atanır. Daha sonra 1 ile bu üst sınır arasındaki tamsayıları üretecek olan döngü açılır ve döngü içinde ardışık toplama işlemi gerçekleştirilir.



Eğer klavyeden 'N' için 5 girilirse; programın çalışması Tablo 3.2'deki gibi olur.

Tablo 3.2: Örnek-3.1'in çalışmasında oluşan değerler

I	Eski T	Yeni T
1	0	0+1=1
2	1	1+2=3
3	3	3+3=6
4	6	6+4=10
5	10	10+5=*15*

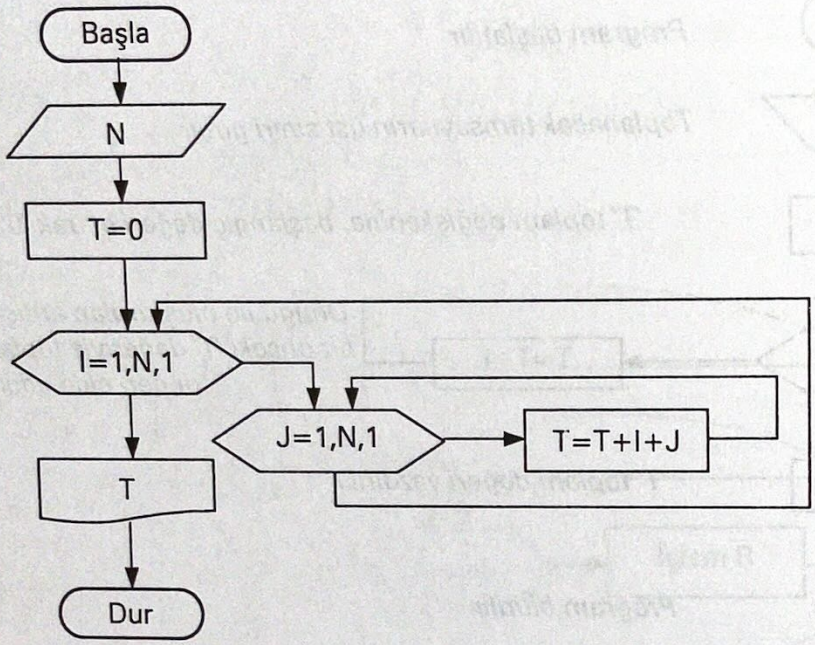
Eğer işlem kutusundaki 'T=0' yerine 'F=1' ve döngü içindeki 'T=T+I' yerine 'F=F*I' yazılırsa, bu program; klavyeden girilen pozitif bir tamsayının faktöriyelini hesaplar (Tablo 3.3).

Tablo 3.3: Faktöriyel alma işleminde oluşan değerler

I	Eski F	Yeni F
1	1	1*1=1
2	1	1*2=2
3	2	2*3=6
4	6	6*4=24
5	24	24*5=*120*

Örnek-3.2: İç içe iki döngü bulunan bir akış diyagramının temsil ettiği işlemlerin incelenmesi.

İç içe döngülerde; dıştaki döngünün her çalıştırılmasında içteki döngü tekrarlama sayısınca işlenir. Yani her bir 'I' değerinde 'J' döngüsü tekrar baştan başlar ve içindeki 'T=T+I+J' işlemi yapılır.

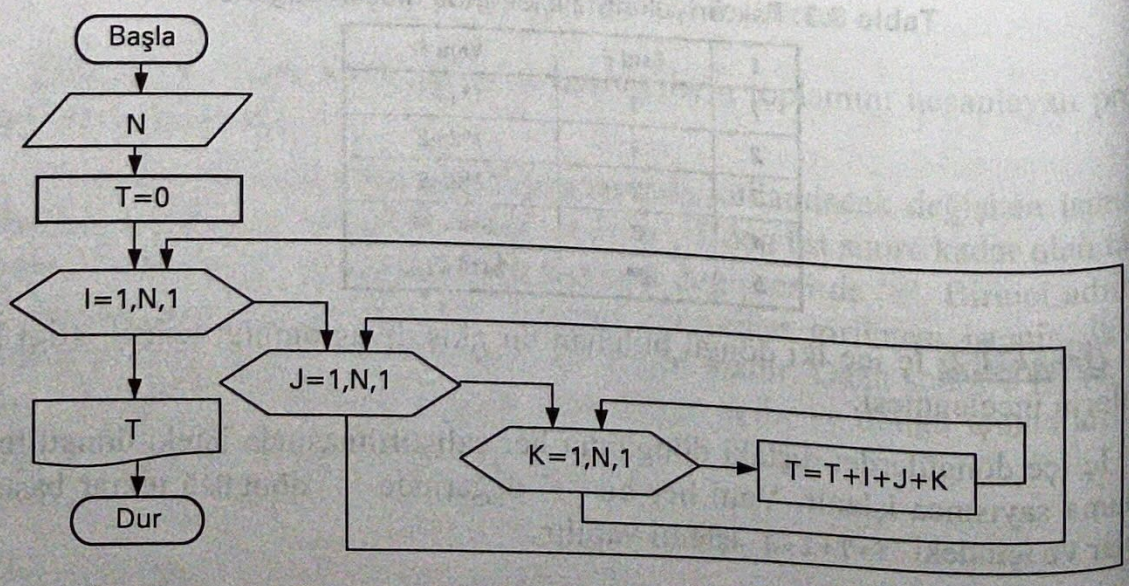


Akış diyagramının N=3 değeri için çalışması, Tablo 3.4'teki gibi olur. Tablo-3.4'te görüldüğü gibi, 'I' nın her değerinde ilk önce en içteki döngü (J) tamamlanmakta, daha sonra dış döngüye (I) çıkılmaktadır.

Tablo 3.4: Örnek-3.2'nin çalışma prensibi

I	J	Eski T	Yeni T
1	1	0	0+1+1=2
1	2	2	2+1+2=5
1	3	5	5+1+3=9
2	1	9	9+2+1=12
2	2	12	12+2+2=16
2	3	16	16+2+3=21
3	1	21	21+3+1=25
3	2	25	25+3+2=30
3	3	30	30+3+3=36

Örnek-3.3: İç içe üç döngü bulunan bir programın akış diyagramının incelenmesi.



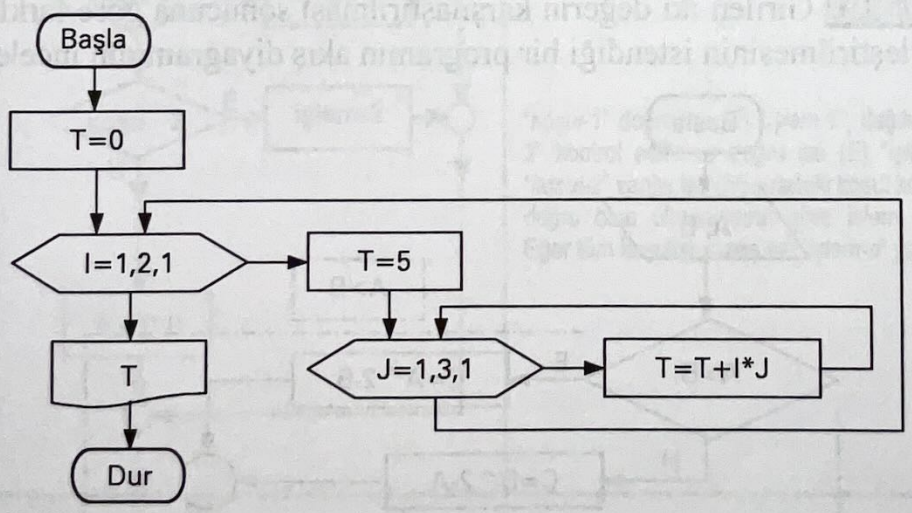
Böyle bir programda her bir 'I' değeri için 'N' sayıda 'J' döngüsü ve her bir 'J' değeri için de 'N' sayıda 'K' döngüsü tekrarlanır.

İç içe üç döngüsü bulunan akış diyagramında N=2 değeriyle programın çalışması durumunda, Tablo 3.5'teki değerler oluşur. Tablo 3.5'ten de görüldüğü gibi 'I' döngüsü 'J' döngüsünü, 'J' döngüsü de 'K' döngüsünü başlatmaktadır. En içteki 'K' döngüsü tamamlandığında bir üst döngü olan 'J'ye çıkılmakta ve 'J' döngüsü tamamlanmadıysa yeniden 'K' döngüsünü başlatmaktadır. Yani iç döngüden çıkıldığında bir üst döngü tamamlanmadıysa, sıradaki değeriyle içteki döngüyü baştan tekrar çalıştırmaktadır.

Tablo 3.5: Örnek-3.3'teki akış diyagramının çalışma prensibi

I	J	K	Eski T	Yeni T
1	1	1	0	0+1+1+1=3
1	1	2	3	3+1+1+2=7
1	2	1	7	7+1+2+1=11
1	2	2	11	11+1+2+2=16
2	1	1	16	16+2+1+1=20
2	1	2	20	20+2+1+2=25
2	2	1	25	25+2+2+1=30
2	2	2	30	30+2+2+2=36

Örnek-3.4: Aşağıdaki akış diyagramının sonucunu bulunuz.



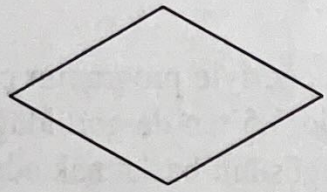
Tablo 3.6: Örnek-3.4'teki akış diyagramının çalışma prensibi

I	J	Eski T	Yeni T
1	1	5	5+1*1=6
1	2	6	6+1*2=8
1	3	8	8+1*3=11
2	1	5	5+2*1=7
2	2	7	7+2*2=11
2	3	11	11+2*3=17

Not

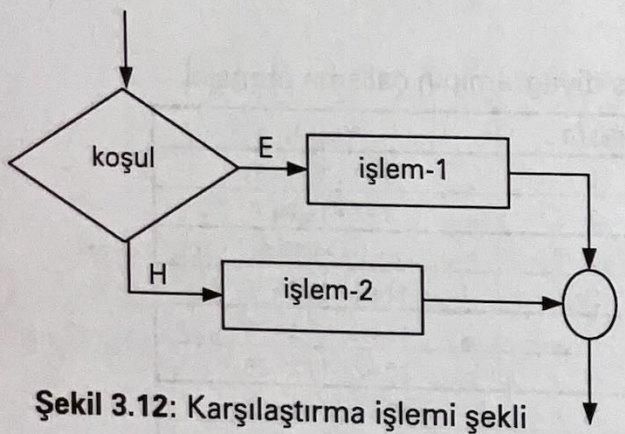
Döngülerden iç döngü tamamlanıp bir üst döngüye çıkıldığında, eğer üst döngü tamamlanmadıysa sıradaki değeriyle içteki döngüyü yeniden başlatmaktadır.

E. Karar (Karşılaştırma)



Şekil 3.11: Karar (karşılaştırma) şekli

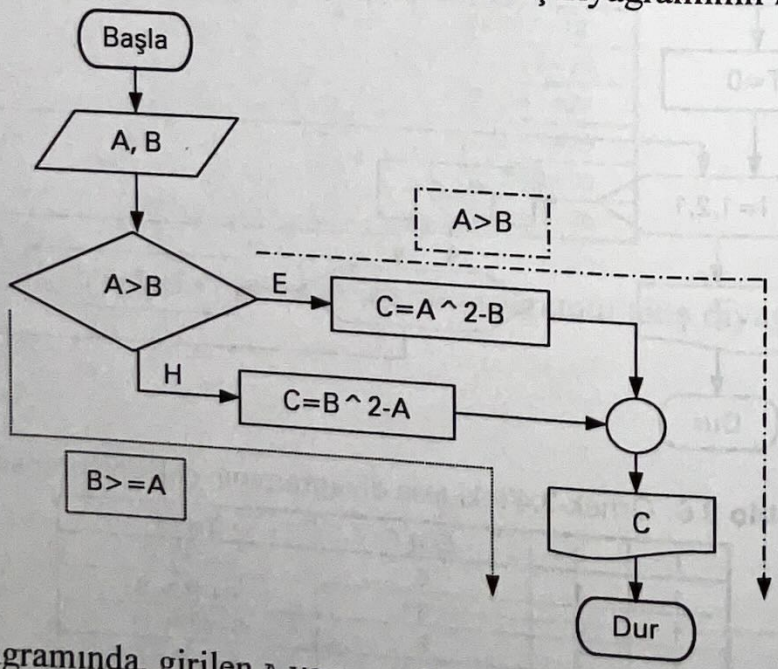
Karar verme (kontrol etme, karşılaştırma) işlemlerini temsil eden şekildir (Şekil 3.11). Koşul veya mantıksal operatörlerle bağlı koşullar, şeklin içine yazılır.



Şekil 3.12: Karşılaştırma işlemi şekli

Karşılaştırma işleminin en genel kullanım biçimi, Şekil 3.12'de verilmektedir. Eğer koşul doğru ise (*Evet*) '`işlem-1`' yapılırken, yanlış ise (*Hayır*) '`işlem-2`' yapılır. İşlemlerden sonra, program akışı bağlantı noktasında birleşerek alt satırlara gider. İşlem; sadece koşulun doğru (veya yanlış) olduğu durumda mevcutsa, diğer işlem kutusunu çizmeye gerek yoktur (Tablo 3.7).

Örnek-3.5: Girilen iki değerin karşılaştırılması sonucuna göre farklı işlemlerin gerçekleştirilmesinin istendiği bir programın akış diyagramının incelenmesi.



Akış diyagramında, girilen `A` ve `B` sayılarından:

- `A` büyükse; $C=A^2-B$ hesaplanıp, işlem sonucu yazdırılır.
- `B` büyük veya eşitse; $C=B^2-A$ hesaplanıp, işlem sonucu yazdırılır.

Örneğin; `A=5`, `B=4` için ekrana '`21`' değeri; `A=3`, `B=6` için de '`33`' değeri yazılır.

Tablo 3.7: Karşılaştırma ifadelerinin genel kullanım şekilleri

Form	Akış diyagramı gösterimi	Açıklama
Yarım form	koşul, E, H, işlem	"*koşul*" doğru ise (*E*) "*işlem*" yapılır, değilse (*H*) akış devam eder.
Tam form	koşul, E, H, işlem-1, işlem-2	"*koşul*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*işlem-2*" yapılır.
Çok koşullu form	koşul -1, E, H, koşul -2, E, H, işlem-1, işlem-2, işlem-n	"*koşul-1*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*koşul-2*" kontrol edilir ve doğru ise (*E*) "*işlem-2*" yapılır. "*koşul-2*" yanlış ise (*H*) sıradaki koşul kontrol edilir ve doğru olup olmamasına göre işlem devam eder. Eğer tüm koşullar yanlış ise "*işlem-n*" yapılır.

Örnek-3.6: Klavyeden girilen bir sayının mutlak değerini ekrana yazdıran programın akış diyagramının incelenmesi.

Herhangi bir sayının mutlak değeri; eğer sayı 0'dan küçük ise "-1" ile çarpılarak, değilse doğrudan alınarak elde edilir. Yani herhangi bir x sayısının $|x|$ mutlak değeri

$$|x| \Rightarrow \begin{cases} x < 0 \quad , \quad |x| = -1x \\ x \geq 0 \quad , \quad |x| = x \end{cases}$$

şeklinde matematiksel olarak ifade edilebilir.

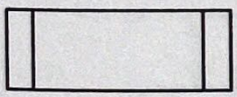
Örnek-3.7: Tablo 3.8'de yazdırma/çıktı örnekleri yer almaktadır.

Tablo 3.8: Örnek yazdırma işlemleri

Örnek şekil	Açıklama
"Uludağ"	Tırnak içindeki 'Uludağ' ismini aynen yazdırır. *Sabit ifadeyi yaz ve alt satıra geç*
x	'x' değişkeninin içeriğini yazdırır ve imleç, bir alt satıra geçer (imleç satırbaşı yapar). *Değişken içeriğini yaz ve alt satıra geç*
y ,	'y' değişkeninin içeriğini yazdırır ve imleç, aynı satırda kalır. Daha sonraki yazdırma işlemleri – herhangi bir konumlandırma ve yönlendirme yoksa – kalınan yerden devam eder. *Değişken içeriğini yaz ve aynı satırda kal*
"Bursa" , a	Tırnak içindeki 'Bursa' ismini ve 'a' değişkeninin içeriğini yazdırır. *Sabit ifade ve değişken içeriğini yazarak alt satıra geç*

G. Önceden tanımlı işlem

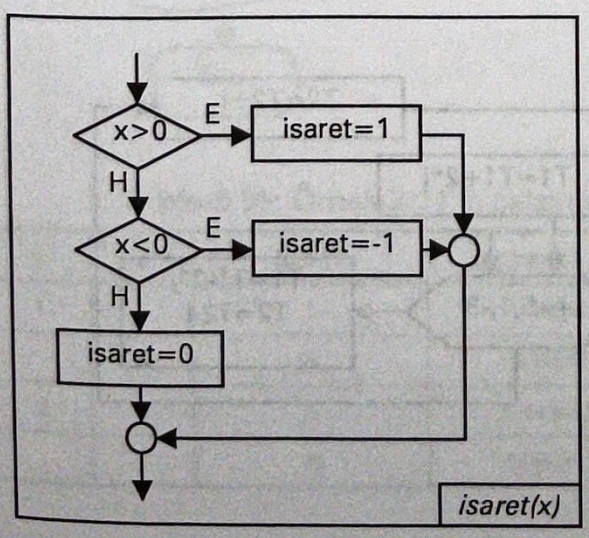
Büyük programlar, birçok küçük programın (modülün) birleşiminden meydana gelmektedir. Benzer şekilde küçük programlarda da bazı işlemler ayrı olarak tanımlanıp (alt program, fonksiyon vb.) kullanılmaktadırlar. Bu şekilde bazı işlem veya işlemlerin yapılmasını defalarca kodlamak yerine, bir defa kodlayarak parametresiz veya parametreli olarak kullanılabilmektedir. Böylece programın işlevselliği, modülerliği, hızı gibi birçok özelliği olumlu yönde etkilenmektedir.

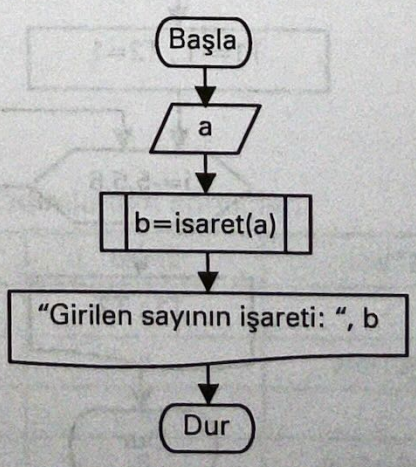


Şekil 3.13 Önceden tanımlı işlem

Önceden tanımlı işlemler, isimleri ve varsa parametreleriyle birlikte Şekil 3.13'tekinin içine yazılır.

Örnek-3.8: Bir sayının işaretini (pozitif, negatif veya sıfır olup olmadığını) bulan alt program yazılıp ana programda rahatlıkla aşağıdaki gibi kullanılabilir.





H. Bağlantı

Şekil 3.14 Bağlantı şekli

Bağlantı şekli (daire) (Şekil 3.14), genel anlamda işlem akışlarını birleştiren bir yer olup aşağıdaki amaçlar için kullanılır:

i. Farklı yerlere dallanan işlem akışlarını toplamak.

ii. Akış diyagramı bir sayfaya sığmadığı zaman diğer sayfadaki akış diyagramı ile bağlantı kurmak.

iii. Parça parça çizilen akış diyagramları arasında bağlantı kurmak.

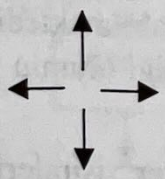
Şekil 3.15 Bağlantı yapma

Parça parça program yazılması veya diyagramın çizilen yere sığmadığı durumlarda; bağlantı yapılacak uçlara bağlantı şekli çizilir ve içine aynı harf/karakter dizisi veya rakam/sayı yazılır (Şekil 3.15).

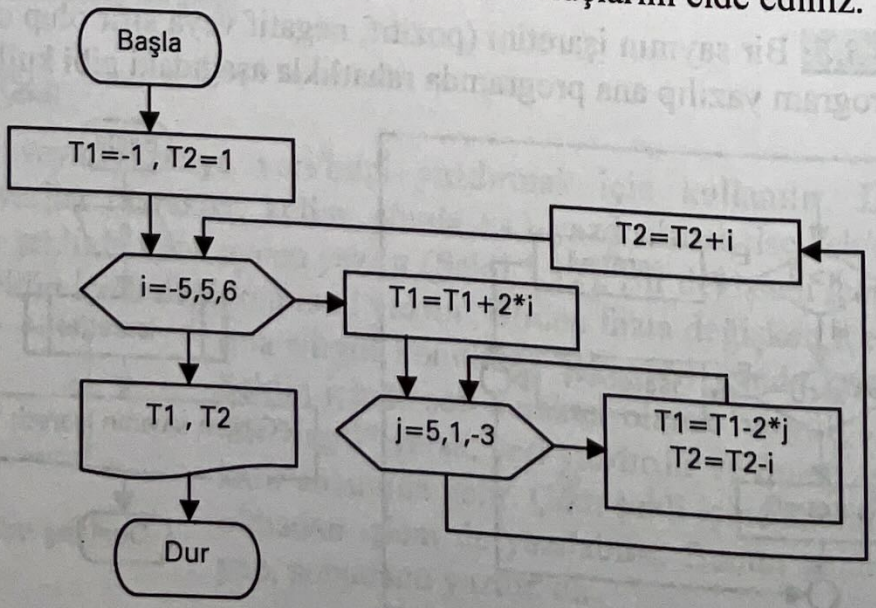
Not

Genel olarak algoritmalarda/akış diyagramlarında/programlarda işlem akışı adım adım ve doğrusal bir şekilde olmalıdır.

I. İşlem Akış Yönleri



Şekil 3.16 İşlem akış yönü şekilleri

Akış diyagramlarında işlem akışının hangi yönde olduğunu gösteren oklardır. İşlem akış yönüne göre uygun olan ok kullanılır (Şekil 3.16).

Örnek-3.9: Aşağıdaki akış diyagramının sonuçlarını elde ediniz.