BLM1011 BİLGİSAYAR BİLİMLERİNE GİRİŞ

GR.2, 3(ONLİNE)

2021-2022 GÜZ YARIYILI

DR.ÖĞR.ÜYESİ GÖKSEL BİRİCİK

Algoritmaya Giriş

Bilgisayar için Bilgi Nedir?

Gerçekleri/doğruları biz biliyor ya da çıkarabiliyoruz. Bunları ifade edebiliyoruz.

- Tanımlayıcı bilgi (Declarative Knowledge)

Bilgisayar ise bunları bilemez.

- Buna karşın, işlemcinin komut kümesinde tanımlı eylemleri çok yüksek hızda ve hatasız çalıştırabilir.

Tanımlayabildiğimiz bilgileri, sonucu elde etmek üzere gerçekleştirilecek şekilde kesin tanımlar haline ifade edebilmemiz gerekir.

- Buyurucu bilgi (Imperative Knowledge)

Algoritma Nedir?

Tanımlayıcı bilgiyi, bilgisayarın işletebileceği buyurucu bilgi haline dönüştürerek ifade etmemiz gerekir.

- Bilgisayarlı hesaplama işlemlerinin hepsinin özünde bu temel kavram yer alır.

Algoritma: Bir problemi çözmek için takip edilecek sonlu sayıda adımdan oluşan çözüm akışının ifadesi.

Bir problemin mantıksal çözümünün adım adım nasıl gerçekleştirileceğinin tanımlanması.

- Sözle ifade edebiliriz.
- Herkesin aynı anlamı çıkarması için tanımlı sembollerden oluşan akış diyagramlarını kullanabiliriz.

Algoritma, belirli bir problemi çözmek için art arda uygulanacak **kesin** direktiflerden oluşan **sonlu** bir kümedir.

Algoritma, mantıklı ve doğru tüm girdiler için **doğrudur** ve doğru çıktıyı sonlu bir zamanda üretir.

Bunun dışındaki durumlar algoritma değildir.

Algoritma’nın Kökeni

Ebû Ca'fer Muhammed bin Mûsâ el-Hârizmî

- (d. 780, Harezm (Horasan – Özbekistan - ö. 850, Bağdat)

Pers gökbilimci ve matematikçi, Bağdat’ta yaşamış

825 yılında, Arapça «Al-jabr»

«Hisab al-jabr w’al-muqabala»

“Hint rakamları ile hesap”

1300lerde latinceye çeviri:

“Algoritmi de numero Indorum”

Tanımdan Direktife Algoritma Oluşturma

Biz gerçekleri tanımlayabiliriz. Karekök alma deklarasyonu: y*y = x (Çıktının karesi girdiye eşittir)

Bilgisayara bunu adım adım, komut setini kullanarak yapmayı anlatmamız gerekir.

0. BAŞLA.
1. x'i oku.
2. Eşik değeri e'yi oku.
3. Rasgele bir r pozitif tamsayısı oluştur.
4. (r*r – x) > e olduğu sürece:
   4.1. r değerini, r ve x/r değerlerinin ortalaması ile güncelle.
5. r değerini yazdır.
6. DUR.

r	r*r	x/r	(r+x/r)/2
3	9	16/3	4.17
4.17	17.36	3.837	4.0035
4.0035	16.0277	3.997	4.000002

Algoritma Özellikleri

Unambiguous (**Anlaşılırlık**): Adımları, giriş ve çıkışları net ve tek bir anlam içermelidir.

Input (**Giriş**): 0 veya daha fazla iyi tanımlanmış girdilere sahip olmalı

Output (**Çıkış**): 1 veya daha fazla iyi tanımlanmış çıktılara sahip olmalı

Finiteness (**Sonlandırılma**): Belirli bir admımdan sonra sonlandırılabilmeli

Feasibility (**Yapılabilirlik**): Mevcut imkanlarla yapılabilir olmalı

Independent (Bağımsızlık): Herhangi bir programlama kodundan bağımsız olmalı

- Algoritma, bir programlama dilinde (Java, C++, C# gibi) ifade edildiğinde **program** adını alır.

Algoritma Tasarımı

Problem çözümünün 4 adımı (George Polya)

- 1. Problemi anlamak
- 2. Çözüm planı oluşturmak
- 3. Planı uygulamak
- 4. Sonucu değerlendirmek

Algoritma tasarlarken ilk iş, ***problemi*** makul (anlaşılır, açıklanabilir) bir şekilde tanımlamaktır.

Örnek: Ön sıradaki en yaşlı öğrenciyi bulalım.

Problem Tanımı:

- **Ön Koşul**: ön sırada oturan öğrenciler olmalı
- **Hedef**: En yaşlı öğrenciyi bulmak
  - Herkes gerçek doğum gününü verecek.
  - Aynı gün doğanlar aynı yaştadır
  - Birden fazla en yaşlı varsa, herhangi birini bildirmek yeterlidir.

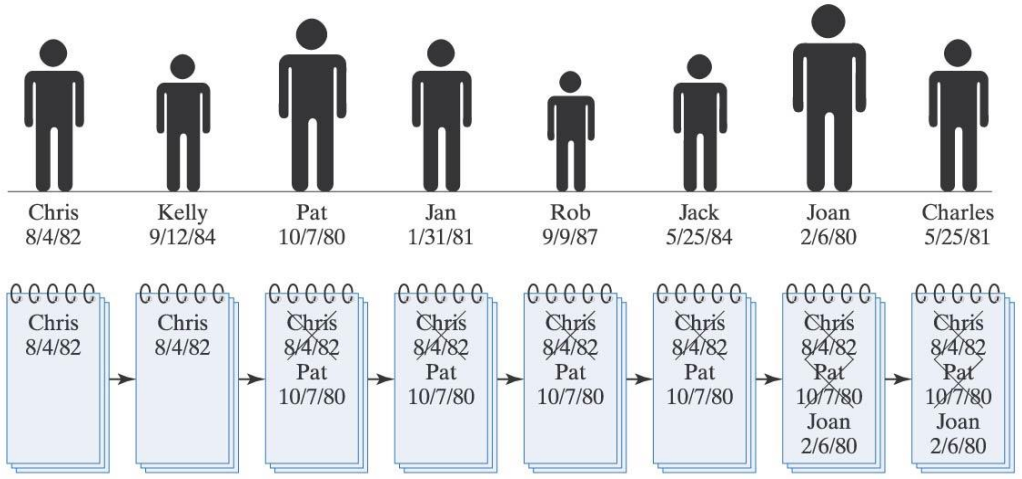
Çözüm – 1

1. Sol baştan başla

2. ilk öğrenciye adını ve yaşını sor, tahtaya yaz.

3. Sırada öğrenci olduğu sürece yap:
   - 3.1. Öğrenciye adını ve yaşını sor.
   - 3.2. Yaş tahtadakinden büyükse, sil ve yeni bilgiyi yaz.

4. Tahtadaki ismi ve yaşını oku.

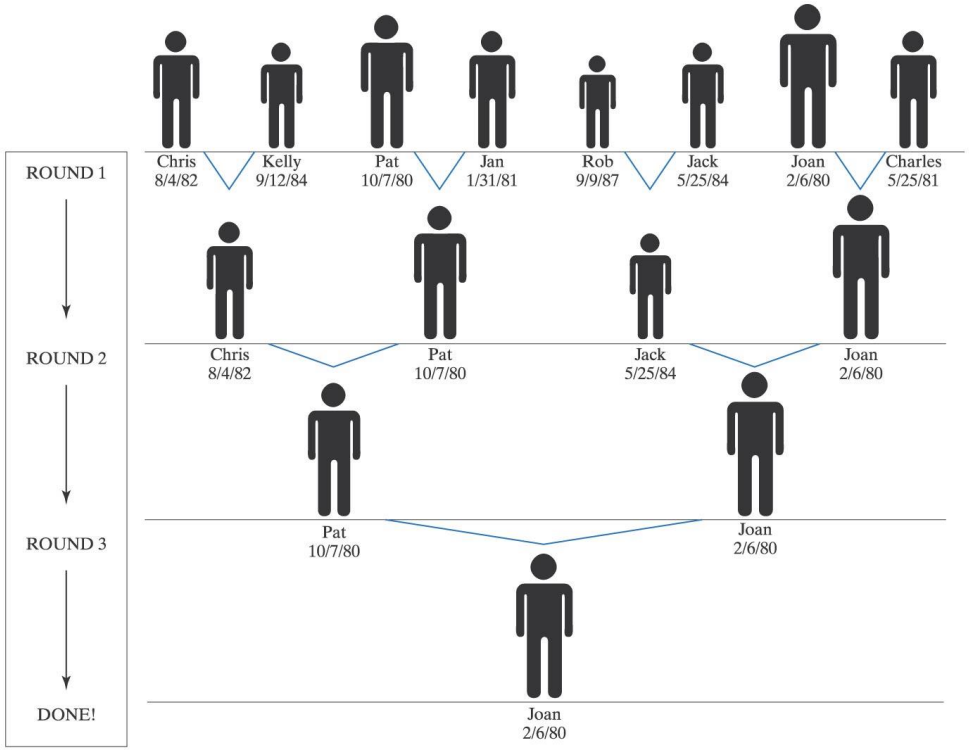


Çözüm – 2

0. Başla

1. Ön sırada birden fazla öğrenci olduğu sürece yap:
   - 1.1. Öğrencileri soldan sağa ikişerli olacak şekilde grupla.
   - 1.2. Sonda tek öğrenci kalırsa, onu tut.
   - 1.3. Her öğrenci çiftinin yaşlarını karşılaştır.
   - 1.4 Genç olanı ön sıradan kaldır.

2. Kalan öğrencinin adını ve yaşını sor, tahtaya yaz.



Sonucu Değerlendirmek – Algoritma Analizi

Hangi çözümün daha «iyi» olduğunu belirlemek her zaman kolay olmayabilir.

- Kolay uygulanabilir olmasını isteyebilirsiniz.
- Sadece belirli bir girdi/ortam koşulunda çalışmasını bekleyebilirsiniz.
- Daha hızlı olan daha iyi olabilir.
- Daha az yer harcayan tercih edilebilir.

Algoritma-1'de yapılması gereken iş, ön sırada oturan öğrenci sayısı ile doğru orantılıdır.

- Öğrenci sayısı iki katına çıkarsa, iş de iki katına çıkar.

Algoritma-2'de çiftlerin sorgularını aynı anda yapma imkanımız da olabilir.

- Yapılan iş, sorgulanan kişi sayısını kaç kere azalttığımız ile orantılıdır.
- Her adımda yarıya düştüğümüze göre, öğrenci sayısının 2 tabanındaki logaritması ile orantılıdır.
- Öğrenci sayısı 2 katına çıkarsa, süre 1 fazla karşılaştırma kadar artar.

Sayısal Karşılaştırma

Her karşılaştırma 5 saniye sürsün.

Algoritma-1 :

- 100 öğrenci için 5*100 = 500 saniye
- 200 öğrenci için 5*200 = 1000 saniye
- 400 öğrenci için 5*400 = 2000 saniye

- 1,000,000 öğrenci için 5*1000000 = 5,000,000 saniye

Algoritma-2:

- 100 öğrenci için $5*\log_2 100 = 35$ saniye
- 200 öğrenci için $5*\log_2 200 = 40$ saniye
- 400 öğrenci için $5*\log_2 400 = 45$ saniye

- 1,000,000 öğrenci için $5*\log_2 1000000 = 100$ saniye

Algoritmaları İfade Etmek

Algoritmayı İfade Etmek

Etken ve buyuran cümleler ile düz yazı şeklinde ifade edebiliriz.

- Okuyanlar farklı anlayıp farklı yorumlayabilir.

Sözde-kod ile akışı tanımlayabiliriz.

- Günümüzde sıklıkla –özellikle detaylardan kaçınınca- kullanılıyor.

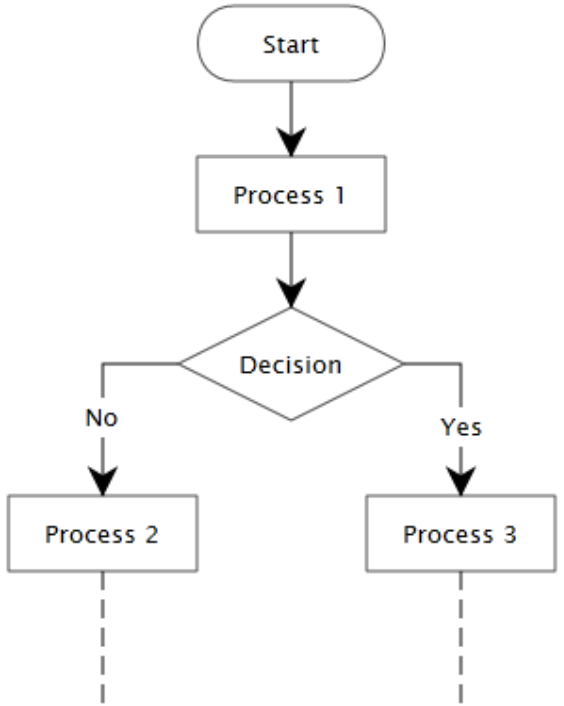
Akış diyagramları ile gösterebiliriz.

- Görsel olarak akışı en temiz hali ile göstermiş oluruz.
- Semboller evrensel olduğu için her okuyan aynı ifadeyi anlar.

Algoritmayı İfade Etmek-Akış Diyagramı

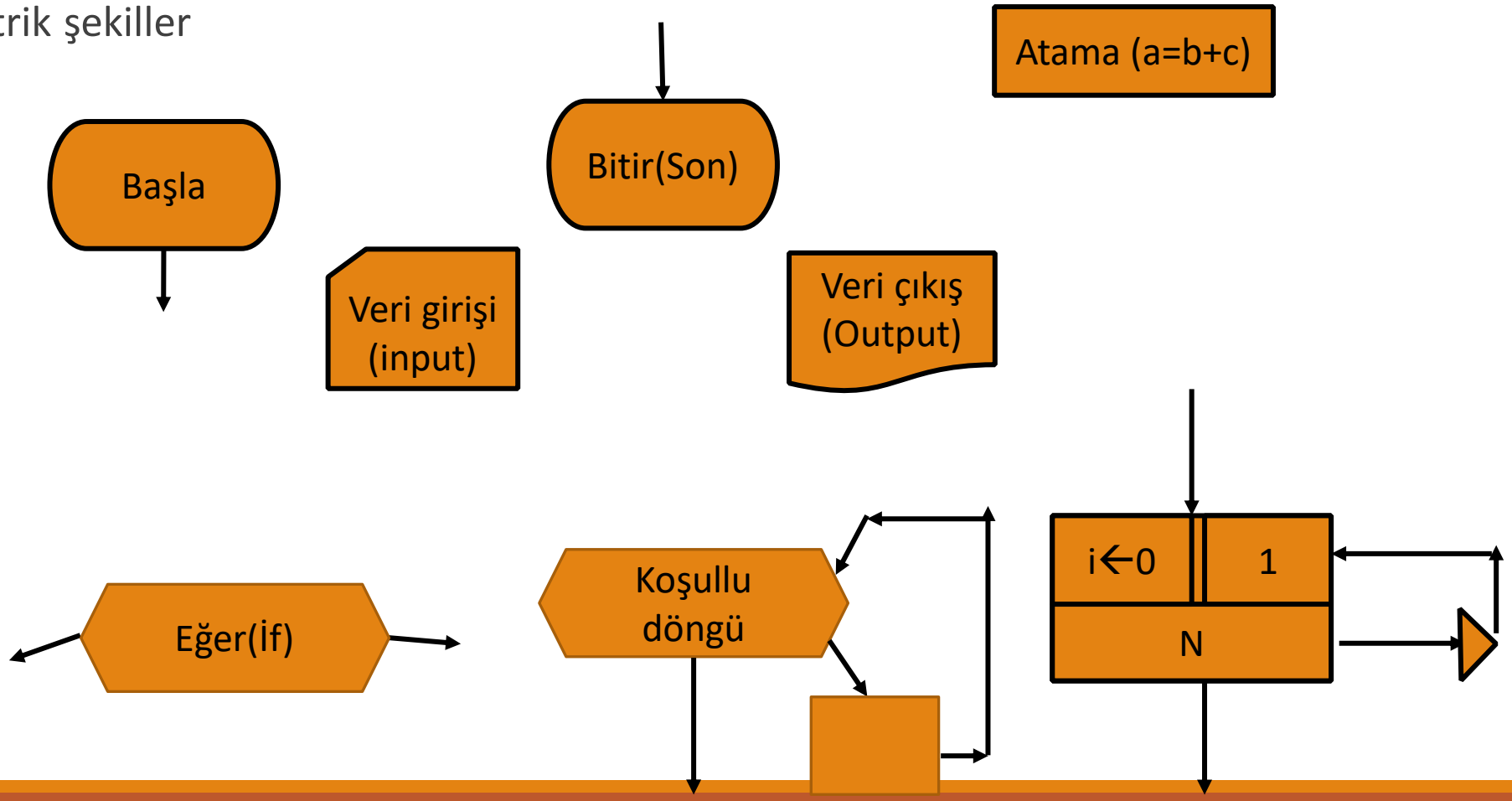
Akış diyagramı elemanları

Bağlantılar



Algoritmayı İfade Etmek-Akış Diyagramı

Geometrik şekiller



for (int i=0; i<N; i++)

Akış Diyagramı Bileşenleri– I

Başla - Bitir (Start - Stop)

Okuma (Read / Inputs)

Bağlantılar (Connections)

- Oklar, çemberler (arrows, circles)

İfadeler/İşlemler (Statements/Process)

Kararlar (Decisions)

- if, case/switch, ...

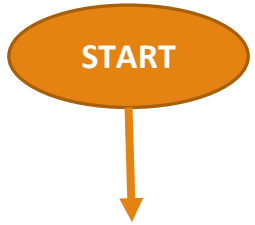
Döngüler (Loops)

- for, while, do while ...

Akış Diyagramı Bileşenleri– II

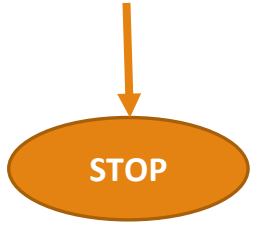
BAŞLA / START

- Algoritmanın başlangıcını gösterir
- Tüm akış diyagramlarının bir başlangıcı olmalıdır.



BİTİR / STOP

- Algoritmanın sonlandığı yeri gösterir
- Tüm akış diyagramlarının bir sonu olmalıdır.



Akış Diyagramı Bileşenleri– III

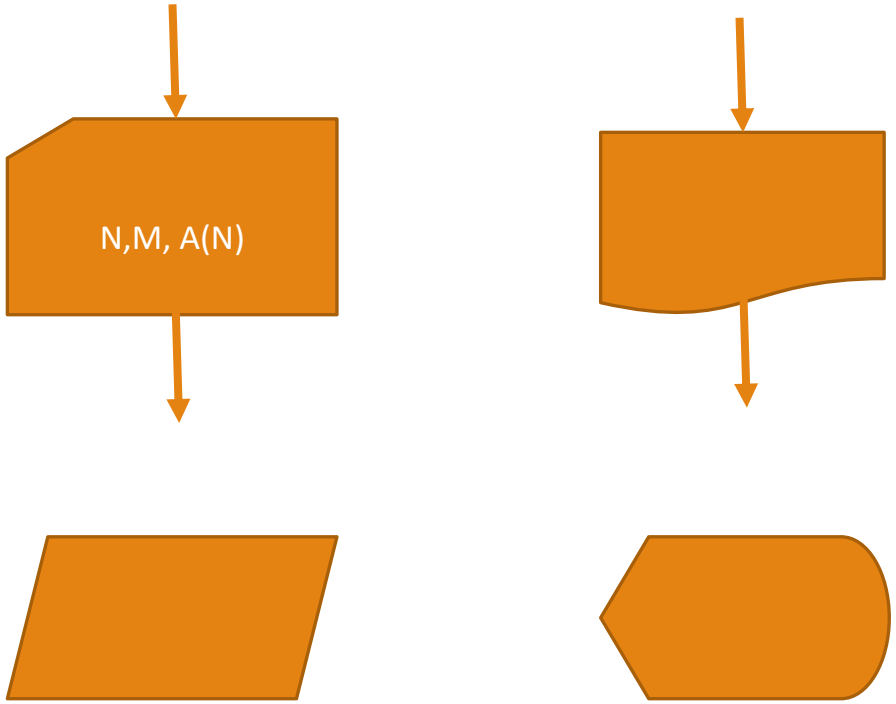
Girdi

- Kullanıcıdan alınan değişkenleri gösterir.
- Her değişken virgül ile ayrılarak yazılır.

Dizi okuma

- 1.seçenek: for döngüsü içinde girdi kullanın
- 2.seçenek: A(N) notasyonunu kullanın
  - **A(N)'den önce N'i okumalısınız!**

Çıktı



Akış Diyagramı Bileşenleri– IV

Oklar

- Akıştaki iki elemanı birbirine bağlar
- Akışın yönünü gösterir
- **Eğri büğrü değil, köşeli çizilir.**

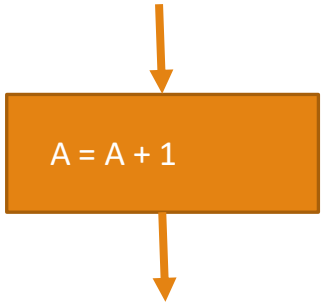
Çemberler

- Bağlantıları basitleştirmek / okunabilir kılmak için
  - İçiçe döngüler, içiçe kontroller vs
- **Numaralandırmalara dikkat edin!**

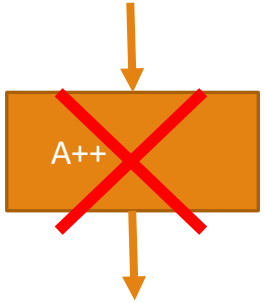
Akış Diyagramı Bileşenleri– V

İfadeler / İşlemler

- Aşağıdaki ifadelerden en az birini içerir:
  - Aritmetik işlem, atama, …



- Algoritmalar programlama dillerinden bağımsız tasarlanmalıdır!
  - Dile özel operatör KULLANILMAMALIDIR
  - Dile özel değişken KULLANILMAMALIDIR

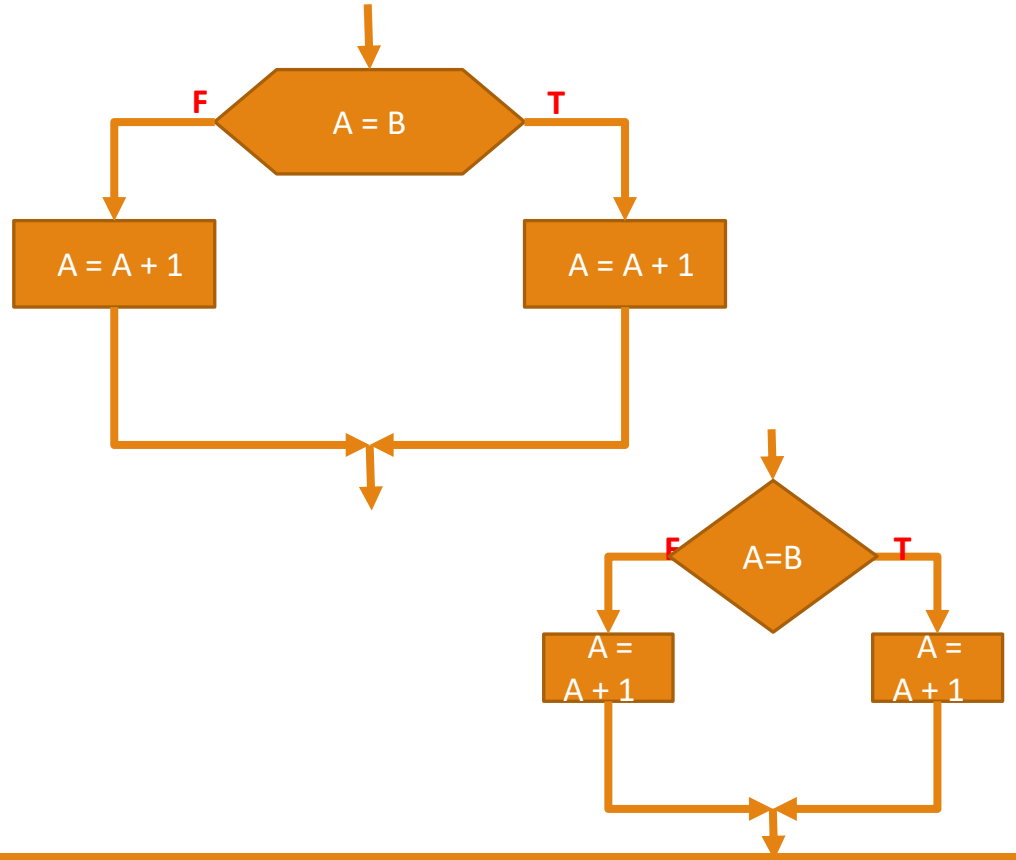


Akış Diyagramı Bileşenleri– VI

IF İfadesi

- Her zaman Doğru **(T)**rue ve Yanlış **(F)**alse kollarını belirtin.
- Her if ifadesinde MUTLAKA doğru (T) dalı olmalıdır.
- Yanlış (F) dalı isteğe bağlıdır.

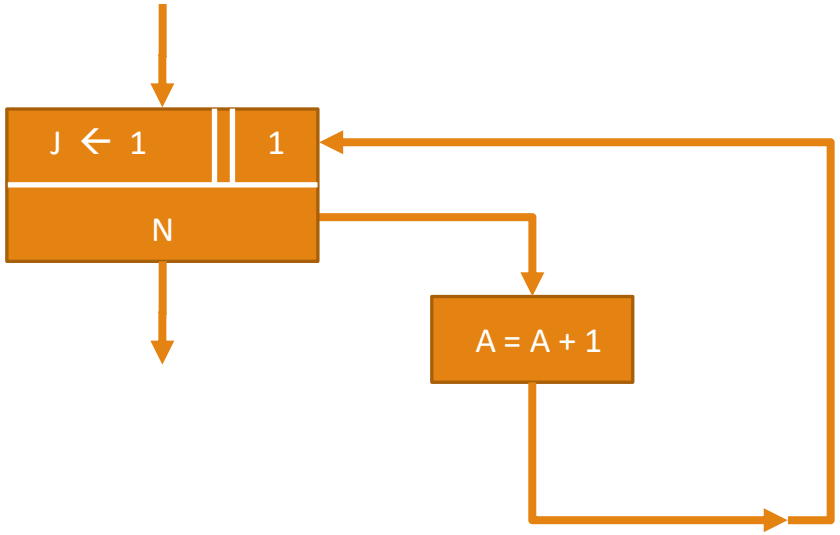
Algoritmadaki tüm IF ifadelerinde T/F dallanmalarını aynı tarafta kullanmak tercih sebebidir.

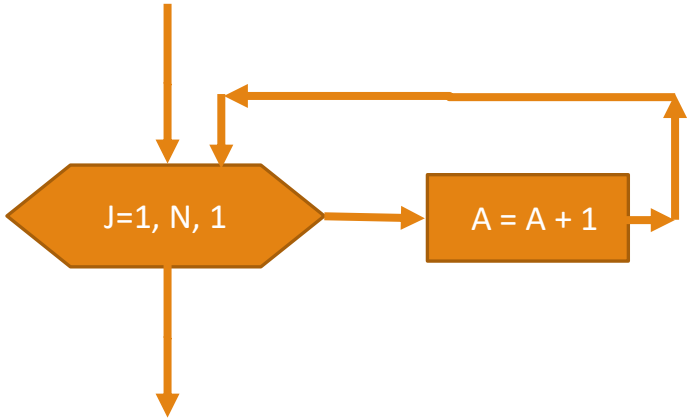


Akış Diyagramı Bileşenleri– VIII

FOR döngüleri

- Belirli sayıda iterasyona sahip döngüler için kullanılır.
- Üç parçadan oluşur
  - İlk koşul (başlangıç)
  - Son koşul (durma)
  - Adım (+/-)

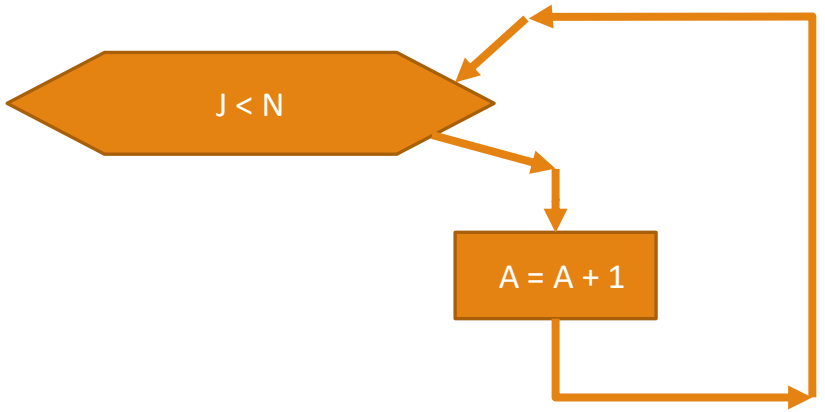




Akış Diyagramı Bileşenleri– IX

While döngüleri

- Bilinmeyen sayıda iterasyona sahip döngüler için kullanılır.
  - Bir şartın sağlandığı sürece işletilmesi esasına dayanır



Algoritmayı İfade Etmek-Sözde Kod

Yapısal programlama modelini ilk önerdiğinde, Edsger Dijkstra herhangi bir algoritmanın sadece üç programlama yapısını kullanarak yazılabileceğini belirtti: **dizilim**, **seçim** ve **döngü**.

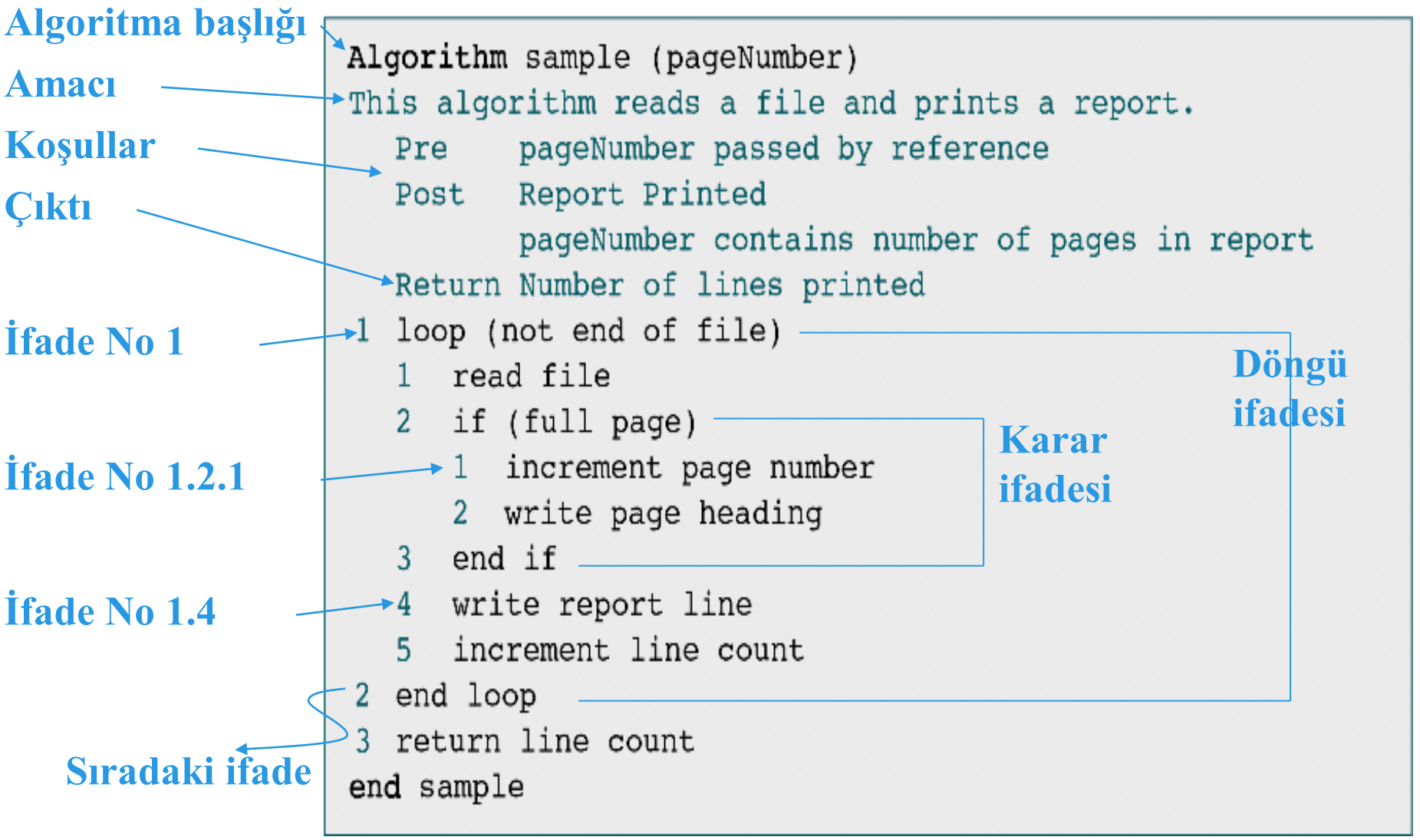
Bir **dizilim**, bir algoritma içindeki yürütme yolunu değiştirmeyen bir veya daha fazla ifadedir.

Bir **seçim** ifadesi bir koşulu değerlendirir ve sıfır veya daha fazla alternatif yürütür.

- Değerlendirme sonuçları hangi alternatiflerin işletileceğini belirler.

Bir **döngü** ifadesi bir kod bloğunu tekrarlar.

Algoritmayı İfade Etmek-Sözde Kod



Algoritmayı İfade Etmek-Sözde Kod

Atama→ genellikle := (Pascal dilindeki gibi) ya da ←

- Max := 1000
- Sum ← Sum + newItem

Eşitlik kontrolü → =

- a = b

Seçim

- if (condition) action1 end if
- if (condition) action1 else action2 end if

Döngü

- while (condition) do statements end while
- do statements while (condition)
- repeat statements until (condition)
- for start to end step increment do statements end for

İndis

- arr[i]
- avg[i+3]

Örnek: Bir dizinin toplamı

Algoritma ToplamHesapla (dizi)

Verilen dizinin elemanlarının toplamını hesaplar

Girdi: n elemanlı tamsayı dizisi

Çıktı dizinin elemanlarının toplam değeri

1 toplam := 0

2 for i := 1 to n do

1 toplam := toplam + dizi[i]

3 endfor

4 return toplam

end ToplamHesapla

Örnek Algoritmalar

Örnekler

1. Verilen N adet sayının toplamını ve ortalamasını yazdıralım.
2. Üç adet tamsayı okunuyor. Bunların ortancası hangisidir? Bulup yazdıralım.
3. N'e kadar olan Fibonacci sayılarını yazdıralım.
4. Verilen bir sayının tüm bölenlerini yazdıralım.
5. Sadece toplama ve çıkarma kullanarak iki sayıyı çarpalım / kalanlı bölelim.
6. Okunan N için $f(x) = \sum_{x=1}^{n} \left(\frac{1}{x^2}\right)$ fonksiyonunun sonucunu hesaplayıp yazdıralım.
7. Okunan iki sayının OBEB'ini bulalım.

Minibüsçü Abi Para Üstü Verecek. Bir Gözü Yolda, Bir Eli Direksiyonda. En Az Sayıda Bozuk Parayı Nasıl Versin?