SAKARYA ÜNİVERSİTESİ

Programlamaya Giriş
HAFTA 2
Algoritma, Sözde Kod ve Akış Diyagramı

Prof. Dr. Cemil ÖZ

Doç. Dr. Cüneyt BAYILMIŞ

Dr. Öğretim Üyesi Gülüzar ÇİT

Konu & İçerik

- Algoritma Nedir? İyi Bir Algoritmada Neler Olmalıdır?
- Büyük Resim – Algoritma Farklı Şekillerde İfadeleri
  - Algoritma, Sözde Kod, Akış Diyagramı
- Algoritma Tasarımında Temel Kavramlar/Yapılar/İşlemler
  - Değişken, Sabit, Sayaç Sorgu, Döngü
  - Matematiksel, Mantıksal ve Karşılaştırma İşlemleri
  - Algoritma Örnekleri
- Akış Diyagramlarında Kullanılan Geometrik Semboller
  - Akış Diyagramı Örnekleri
- Kaynaklar

Algoritma

- Gerçek hayattaki "plan" kelimesinin karşılığı olarak düşünülebilir.
- Bir problemin çözümüne yönelik işlem basamaklarının sıralı biçimde ve sonlu olarak ifade edilmesi
- Bilgisayar dilinde " bir sorunun çözümü için öngörülen işlemlerin mantıksal ve sembolik anlatımı"
- 9. yüzyılda yaşamış Türk-İslam matematikçi ve astronomu Harzemli Mehmet, toplama, çıkarma, ikiye bölme, bir sayının iki katını bulma, denklem çözümü, vb. gibi cebirsel işlemleri açıklayan bir çalışma yaptı. Batılılar, onun bu çalışmalarına Latince "algorismus" yani bugünkü adı ile **algoritma** dediler ve algoritma kavramı ilk defa burada kullanılmış oldu.
- Bir algoritma yazmak için mutlaka bir dile bağımlı kalma zorunluluğu yoktur. Önemli olan yazılan algoritmanın herhangi bir programlama diline uyarlanabilir olmasıdır.

Algoritma…

- **İyi Bir Algoritmada Neler Olmalıdır?**
  - **Etkinlik**
    - Gereksiz tekrarlarda bulunmamalı, diğer algoritmalar içerisinde de kullanılabilir olmalıdır.
  - **Sonluluk**
    - Her algoritma bir başlangıçtan oluşmalı, belirli işlem adımı içermeli ve bir bitiş noktasına sahip olmalıdır. Kısır bir döngüye girmemelidir.
  - **Kesinlik**
    - İşlem sonucu kesin olmalı, her yeni çalıştırmada aynı sonucu üretmelidir.
  - **Giriş/Çıkış**
    - Algoritma giriş (üzerinde işlem yapılacak değerler) ve çıkış (yapılan işlemler neticesinde üretilen sonuç değerler) değerlerine sahip olmalıdır.
  - **Başarım/Performans**
    - Amaç, donanım gereksinimi (bellek kullanımı, vs.), çalışma süresi, vs. gibi performans kriterlerini dikkate alarak yüksek başarımlı programlar yazmak olmalıdır.

Algoritma

- **Algoritmalar Farklı Şekillerde İfade Edilebilir mi?**
  - Algoritmanın metin olarak yazılması
    - Çözülecek problem, adım adım metin olarak yazılır.
  - **Sözde/Kaba Kod** (Pseudo Code)
    - Algoritmanın sözde kodlarla yazılmasıdır.
    - Problemin çözüm adımları komut benzeri anlaşılır metinlerle ifade edilir.
    - Yarı kod yarı metin olarak da adlandırılır.
    - Sözde kod, programlar gibi derlenmez ve işlenmez.
  - **Akış Diyagramı**
    - Problemin çözüm adımları geometrik şekiller ile ifade edilir.

Algoritma…

- **Akış Diyagramı**
  - Algoritmaların özel geometrik şekiller ile gösterilmesidir
- **Sözde Kod**
  - Algoritmaların ifade edilmesinde akış diyagramlarının yanı sıra, konuşma dili ile programlama dili arasında, **sözde kod** (pseudo-code) adı verilen bir araç kullanılır.
  - Programlama dilinden bağımsız olarak yazılan kodlardır, yani programlar gibi derlenmez veya işlenmezler.
  - Her programcı kendi sözde kodunu geliştirebilir . Fakat kişisel sözde kodlar başkaları tarafından anlaşılabilir bir biçimde açık olmalıdır.

Algoritma…

ÖRNEK

- Problem: Klavyeden girilen sayının karesini hesaplayarak ekrana yazdıran programın algoritmasını yazınız.
- Algoritma:

1. **Başla**
2. **Sayıyı (A) gir**
3. **Sayının karesini hesapla (Kare = A*A işlemini yap)**
4. **Sonucu (Kare) yaz**
5. **Dur**

Algoritma…

ÖRNEK…

- Problem: Klavyeden girilen sayının karesini hesaplayarak ekrana yazdıran programın sözde kodunu yazınız.

- Sözde Kod:

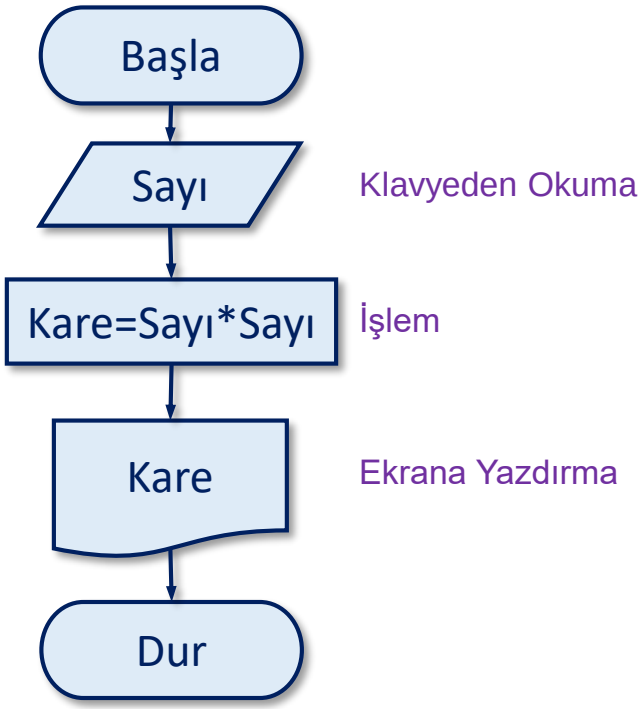
1. **/* Kare alma programı */**
2. **cin >> A**
3. **kare=A*A**
4. **cout << kare**
5. **END**

Algoritma…

ÖRNEK…

- Problem: Klavyeden girilen sayının karesini hesaplayarak ekrana yazdıran programın akış diyagramını çiziniz.

- Akış Diyagramı:



Algoritma Tasarımında Kullanılan Terimler

- **Operatör**
  - İşlemleri belirten yani veriler üzerinde işlem yapma özelliği olan simgelerdir.
  - Örnek:
    - + = >

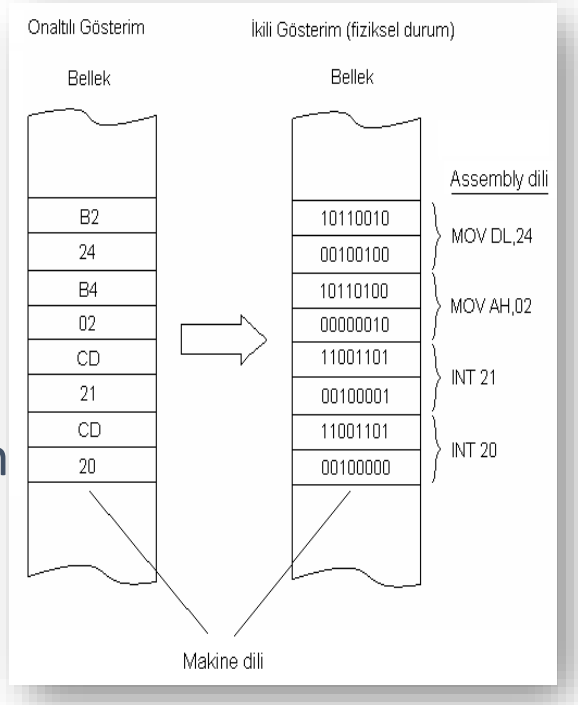
Algoritma Tasarımında Kullanılan Terimler…

- **Tanımlayıcı**
  - Programcı tarafından oluşturulan/verilen ve programdaki değişkenleri, sabitleri (değişmez), sınıf, nesne, özel bilgi tiplerini, alt programları vb. adlandırmak için kullanılan kelimelerdir.
  - Tanımlayıcı, yerini tutacağı ifadeye çağrışım yapmalıdır.

- **Tanımlayıcı kelimeler oluşturulurken uyulması gereken kurallar:**
  - İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir
  - 0-9 arası rakamlar kullanılabilir
  - Simgelerden sadece alt çizgi (_) kullanılabilir
  - Tanımlayıcı isimleri, harf veya alt çizgi ile başlayabilir
  - Rakam ile başlayamaz veya sadece rakamlardan oluşamaz
  - Kullanılan programlama dilinin komutu ya da saklı kelimelerinden olamaz

Algoritma Tasarımında Kullanılan Terimler…

- **Değişken**
  - Verilerin saklandığı bellek alanlarına verilen simgesel isimlerdir.
  - Programın her çalıştırılmasında, farklı değerler alabilen/aktarılabilen bilgi/bellek alanlarıdır.
  - Bir programda birbirinden farklı kaç tane veri tutulacak ise o kadar değişken tanımlanmalıdır.
  - Değişkenleri adlandırma, tamamen programcının isteğine bağlıdır.
  - Değişken adlandırmada tanımlayıcı isimlendirme kuralları geçerlidir.
  - Değişken adının, yerini aldığı ifadeyi çağrışım yapması programın anlaşılırlığı açısından önemlidir.
    - Örnek:
      - Bir kişinin adını tutmak için **Ad** , telefonunu tutmak için **Tel**

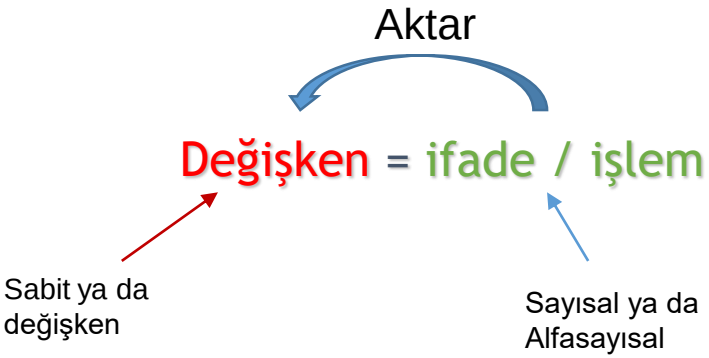


Algoritma Tasarımında Kullanılan Terimler

- **Sabit**
  - Programlardaki değeri değişmeyen ifadelere **sabit** denir.
  - Sabit adlandırmalarında da tanımlayıcı kuralları geçerlidir.
  - Sabitlere değer aktarma
    - Sayısal veriler doğrudan aktarılır.
      - Örnek: SabitKomutu pi = 3.14
    - Alfasayısal (karakter) veriler ise tek/çift tırnak içerisinde aktarılır.
      - Örnek: SabitKomutu ilkharf = 'A'
      - Örnek: SabitKomutu okulAdi = "Sakarya"

Algoritma Tasarımında Kullanılan Terimler

- **Aktarma**
  - Bir bilgi alanına, veri yazma
  - Bir ifadenin sonucunu başka bir değişkende gösterme vb. görevlerde aktarma " = " operatörü kullanılır.
  - Kullanım şekli:



Örnek:

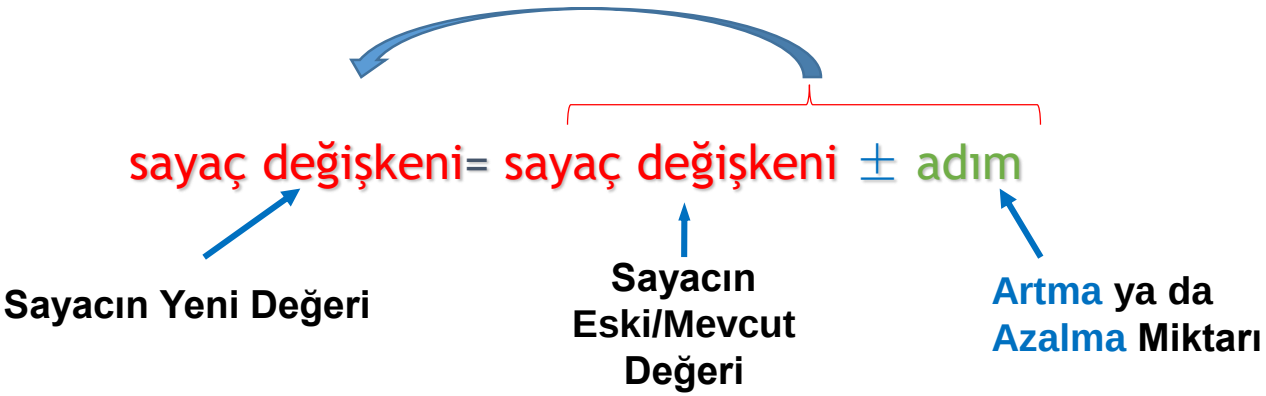
- Sayısal ifade olarak
  - A= 2, B= 3
  - C = A + B
- Alfasayısal ifade olarak
  - A = "Sak" , B = "arya"
  - C = A+B

C=5

C="Sakarya"

Algoritma Tasarımında Kullanılan Terimler…

- **Sayaç / Sayıcı**
  - Belirli sayıda yapılması istenen işlemleri takip etmek için
  - İşlenen ya da üretilen değerlerin sayılması gerektiği durumlarda kullanılır.
  - Örnek: Rasgele oluşturulmuş bir dizideki tek sayıların tespitinde tek sayıların adedini belirlemek için sayaç kullanılır.
  - Kullanım şekli:



Örnek:

- Say = Say + 1

Algoritma Tasarımında Kullanılan İşlemler

Matematiksel (Aritmetik) İşlemler

İşlem	Matematik	Bilgisayar
Toplama	a + b	a + b
Çıkarma	a - b	a - b
Çarpma	a . b	a * b
Bölme	a ÷ b	a / b
Üs Alma	a^b	a ^ b

Algoritma Tasarımında Kullanılan İşlemler

Matematiksel (Aritmetik) İşlemler…

Matematiksel Yazılım	Bilgisayara Kodlanması
a+b-c+2abc-7	*a+b-c+2*a*b*c-7*
$a+b^2-c^3$	*a+b^2-c^3*
$a-\frac{b}{c}+2ac-\frac{2}{a+b}$	*a-b/c+2*a*c-2/(a+b)*
$\sqrt{a+b}-\frac{2}{b^2-4ac}$	*(a+b)^(1/2)-2/(b^2-4*a*c)*
$\frac{a^2+b^2}{2ab}$	*(a^2+b^2)/(2*a*b)*

Algoritma Tasarımında Kullanılan İşlemler…

Matematiksel (Aritmetik) İşlemler…

Sıra	İşlem	Bilgisayar
1	Sayıların Negatifliği	-…
2	Parantezler	(…..)
3	Matematiksel Fonksiyonlar	*cos*, *sin*, *log*, …
4	Üs alma	a ^ b, *pow*, …
5	Çarpma ve Bölme	a * b ve a/b
6	Toplama ve Çıkarma	a + b ve a - b

- **Örnek:**
  - **Matematiksel ifade :**

$$x = a.b/c + d.e^f - g$$

  - **Bilgisayar ifadesi:**

$$x = a * b/c + d * e\hat{}f - g$$

7 2 3 5 4 1 6

Algoritma Tasarımında Kullanılan İşlemler…

- **Karşılaştırma İşlemleri**
  - İki büyüklükten hangisinin büyük veya küçük olduğu,
  - İki değişkenin birbirine eşit olup olmadığı gibi konularda karar verebilir.
  - Karşılaştırma İşlemlerinin Bilgisayar Dilindeki Karşılıkları

Sembol	Anlamı
= veya ==	Eşittir
<> Veya !=	Eşit Değildir
>	Büyüktür
<	Küçüktür
>= veya =>	Büyük eşittir
<= veya =<	Küçük eşittir

- **Örnek:**

  Eğer A > B ise Yaz " A sayısı B'den büyüktür "

Algoritma Tasarımında Kullanılan İşlemler…

- **Karşılaştırma İşlemleri…**
  - Sayısal karşılaştırmalarda doğrudan değerler karşılaştırılır.
    - Örnek: 25 > 15 “25 sayısı 15 ten büyüktür”
  - Karakter olarak karşılaştırmalarda ise karşılaştırma işlemine ilk karakterlerden başlanılarak sıra ile karşılaştırılır.
    - Örnek: a > c “1. karakter alfabetik olarak daha önde”

  - Not: Karakter karşılaştırma işlemlerinde, karşılaştırma karakterler arasında değil, karakterlerin ASCII kodları arasında yapılır. Örneğin, A nın ASCII kod karşılığı 65 tir. a nın ise ASCII kod karşılığı 97 dir. Büyük ve küçük harfler arasındaki ASCII kod farkı ise 32’dir.

Algoritma Tasarımında Kullanılan İşlemler…

Mantıksal İşlemler

Temel Mantıksal İşlem Karşılıkları

İşlem	Komut	Matematiksel Sembol	Anlamı
VE	AND	.	Koşulların hepsi doğru ise sonuç doğrudur
VEYA	OR	+	Koşullardan en az biri doğru ise sonuç doğrudur
DEĞİL	NOT	,	Sonuç koşulun tersidir. 1 ise 0 dır

Mantıksal İşlemlerde Öncelik Sıraları

Sıra	İşlem	Komut
1	Parantez İçindeki İşlemler	(…..)
2	DEĞİL	NOT
3	VE	AND
4	VEYA	OR

Algoritma Tasarımında Kullanılan İşlemler…

- **Mantıksal İşlemler…**
  - Örnek: Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.
  - Burada iki koşul vardır ve bu iki koşulun da doğru olması gerekir. Yani;

EĞER Yaş>23 VE maaş=asgari ücret ise YAZ

1.KOŞUL *2.KOŞUL*

- *YAZ komutu 1. ve 2.koşulun her ikisi de sağlanıyorsa çalışır*

Algoritma Tasarımında Kullanılan İşlemler…

- **Mantıksal İşlemler…**
  - Örnek: Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65 in üzerinde not alanların isimleri istenmektedir.
  - Burada 3 koşul vardır.
    - Bilgisayar dersinden 65 in üzerinde not almış olmak temel koşuldur.
    - Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gerekir.

EĞER Bilg_not>65 VE (TDili_not>65 VEYA YDil_not>65) ismi YAZ

Algoritma Tasarımında Kullanılan İşlemler/Yapılar

- **Sorgu / Seçme**
  - Algoritmada, işlemler genellikle sıralı adımlardan oluşur.
  - Bazı koşul/şart durumlarında işlem sıralarının değiştirilmesi ve diğer bir işlem sırasının seçilmesi yada programın yeni işlem sırasından devam etmesi gerekebilir.
  - İşlem sıralarının değiştirilmesi ya da bazı koşulların sorgulanması için 'EĞER' sorgu deyimi kullanılır.
  - Örnek:

EĞER ORT > 50 GİT ADIM 7

KOŞUL *Gidilecek İşlem Sırası*

Algoritma Tasarımında Kullanılan İşlemler/Yapılar

- **Sorgu / Seçme**
  - Örnek: Dışarıdan girilen iki sayıyı kıyaslayan programın algoritmasını çıkarınız?

1. **Başla**
2. **OKU X, Y**
3. **Eğer X > Y ise YAZ "X BÜYÜK" Git 6**
4. **Eğer X < Y ise YAZ "X KÜÇÜK" Git 6**
5. **YAZ "EŞİT"**
6. **Dur**

Algoritma Tasarımında Kullanılan İşlemler/Yapılar

- **Sorgu / Seçme…**
  - Örnek: Dışarıdan 10'dan büyük 20'den küçük sayı girilene kadar programı devam ettiren algoritmayı yazınız.

1. **Başla**
2. **OKU X**
3. **Eğer (X > 10 VE X < 20) ise Git 6**
4. **YAZ " SAYI 10 İLE 20 ARASINDA DEĞİLDİR"**
5. **GİT 2**
6. **YAZ "SAYI 10 İLE 20 ARASINDADIR"**
7. **Dur**

Algoritma Tasarımında Kullanılan İşlemler/Yapılar

- **Döngü**
  - Bazı işlemleri belirli sayıda tekrar etmede,
  - Belirli bir aralıktaki ardışık değerler ile işlem yapmada döngü kullanılır.
  - Diğer bir deyişle, programdaki belirli işlem bloklarını, verilen sayıda gerçekleştiren işlem akış çevrimlerine DÖNGÜ denir.

- Örnek:
  - **1'den 5'e kadar sayıların toplamı**
  - **Ard arda ekrana 4 defa SAKARYA yazdırma**
  - **1 ile 100 arasındaki çift sayıların ya da tek sayıların toplamı**

Algoritma Tasarımında Kullanılan İşlemler/Yapılar

- **Döngü…**
  - Döngü oluşturma kuralları
    1. Döngü değişkenine başlangıç değeri verilir
    2. Döngünün artma ya da azalma miktarı belirlenir
    3. Döngünün bitiş değeri belirlenir
    4. Eğer Döngü karar ifadeleriyle oluşturulduysa; döngü değişkeni, döngü içinde adım miktarı kadar arttırılmalı/azaltılmalıdır.

  - Örnek: 1-10 arası tek sayıların toplamı hesaplayan programın algoritmasını çıkarınız?

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**
8. **Yaz T**
9. **Dur**

Algoritma Örnekleri - 1

- **Problem:** Klavyeden okunan bir reel sayının karekökünü bulup sonucu ekrana yazan bir algoritmanın tasarlanması.
- **Tasarım:** öncelikle problemin çözümünün matematiksel olarak ifade edilmesi gerekmektedir;
- a, karekökü bulunmak istenen sayı olsun, x değeri a'nın tahmini karekökü ve b değeri ise a'nın gerçek karekökü ile tahmin edilen karekökü arasındaki fark olsun. Bu durumda a aşağıdaki şekilde ifade edilebilir;
  - $a = (x+b)^2 \Rightarrow a = x^2+2xb+b^2$
- Küçük olması beklenen b^2 değeri ihmal edilirse, b değeri yaklaşık olarak hesaplanabilir;
  - $b \cong (a-x^2)/2x$

Algoritma Örnekleri – 1…

- Hesaplanan b değeri kullanılarak a'nın karekköküne daha yakın yeni bir tahmin yapılabilir;
- $x_{i+1} = x_i + b$
  - Burada x_i önceki tahmin, x_{i+1} ise kareköke yakın yeni tahmin değeridir
- Bu şekilde a'nın karekökü girilerek yakınsayan bir iterasyon (tekrarlama) ile bulunabilir.
- a'nın karekökünü yakınsayarak bulan bu iteratif (mutlak hata $|b|$, ε hata değerinden küçük eşit olana kadar işlem tekrar edilecek).

Algoritma Örnekleri – 1…

- Algoritma sözde kodlar ile ifade edildiğinde aşağıdaki şekilde yazılabilir (ifade kolaylığı için x_i yerine x ve x_{i+1} yerine y kullanılmıştır)

A1: Başla	
A2: Oku (a)	// karekökü bulunacak sayıyı a değişkenine oku
A3: Oku (x)	// ilk tahmini karekökü x değişkenine oku
A4: Oku (ε)	// kabul edilebilir hata değerini ε değişkenine oku
A5: $b=(a-x^2)/2x$	// fark (hata) değeri olan b' yi hesapla
A6: y=x+b	// daha yakın yeni karekök değerini (y) hesapla
A7: Eğer $\|b\|\leq\varepsilon$ ise A10'a git	// $\|b\|\leq\varepsilon$ ise iterasyonu durdurmak için A10'a git
A8: x=y	// y yeni karekök değerini x değişkenine ata
A9: A5'e git	// işlemi yeni x tahmini ile tekrarlamak için A5'e git
A10: Yaz (y)	// en son hesaplanan karekök değerini (y) ekrana yaz
A11: Dur	// programı sonlandır

Algoritma Örnekleri – 1…

- Bu algoritmada işlemlerin bir çevrimin içinde tekrarlandığı ve istenilen hassasiyete ulaşıldığında çevrimin dışına çıkılarak işlemin tamamlandığı görülmektedir. Bilgisayar da program işletilirken bir değişkene yeni bir değer verildiğinde eski taşıdığı değerin kaybolacağı not edilmelidir.
- Aşağıda bu algoritmanın nasıl çalıştığı, işlemlerin her tekrarında (çevrimin her adımında) değişkenlerin aldığı değerler bir çizelgede verilerek açıklanmıştır. a, x ve ε değerlerinin sırası ile 31.8, 5.0 ve 0.005 olarak okunduğu kabul edilsin.

Çevrim adım no	a	x	ε	b	y
	31.8	5.0	0.005		
1	31.8	5.0	0.005	0.68	5.68
2	31.8	5.68	0.005	-0.04	5.64
3	31.8	5.64	0.005	-0.0001	5.64

- Üçüncü çevrim adımında |b| değeri ε değeri olan 0.005' den küçük olduğu için yeni karekök değeri hesaplanmaz en son hesaplanan karekök değeri y=5.64 olarak kalır ve işlem sonlandırılır.

Akış Diyagramı

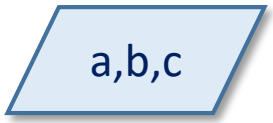
- Akış diyagramı, algoritmaların geometrik şekillerle ortaya konulmasıdır.
- Akış diyagramı, problemin çözümü için yapılması gerekenlerin, başından sonuna kadar geometrik şekillerden oluşan semboller ile ifade edilmesidir.
- Her simge genel olarak yapılacak bir işi veya komutu gösterir.

Akış Diyagramında Kullanılan Semboller

- **Başla**
  - Programın nereden başlayacağını belirtir. Standart olarak her bağımsız algoritmada bir tane bulunur

- **Dur**
  - Programın nerede sonlanacağını belirtir. Birden fazla olabilir. Mümkün ise sadece bir tane dur simgesi kullanılmalıdır.

- **Giriş**
  - Bilgisayara dışarıdan bilgi girişini temsil eder. Bu sembolün içine dışarıdan girilen bilgilerin aktarılacağı değişkenler yazılır.



Akış Diyagramında Kullanılan Semboller…

- **Çıkış**
  - Ekrana veya yazıcıya bilgi göndermeyi temsil eder

- **İşlem**
  - Programın işlemesi sırasında yapılacak işlemleri ifade etmek için kullanılır

- **Karşılaştırma (sorgu)**
  - Verilen koşulun sonucuna göre farklı işlem yapılacağını ifade etmek için kullanılır.

Akış Diyagramında Kullanılan Semboller…

Döngü

- Belirli bir işin veya bir grup işin birden çok yinelenmesi gerektiğinde kullanılır. Döngüdeki çevrim sayısı, döngü sayacı ve sayaç artırımı açıkça yazılır.

Fonksiyon Çağırma

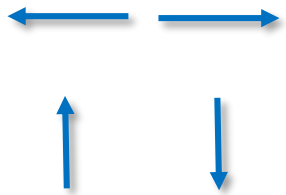
- Daha önce oluşturulmuş bir algoritmanın yazılan algoritma içerisine konulmadan çağrılarak kullanılmasını ifade eder.

Dosyaya Saklama

- Elde edilen bilgilerin bir dosyada saklanması veya dosyadan okunmasını simgeler.

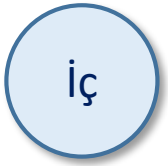
Akış Diyagramında Kullanılan Semboller…

- **Akış Yönü**
  - Bir işlem bittikten sonra akışın nereye yönleneceğini belirler.



- **Bağlantı**
  - Akış diyagramı çizilirken sayfaya sığmama durumunda çizimin başka bir yerden devam etmesi için kullanılır.

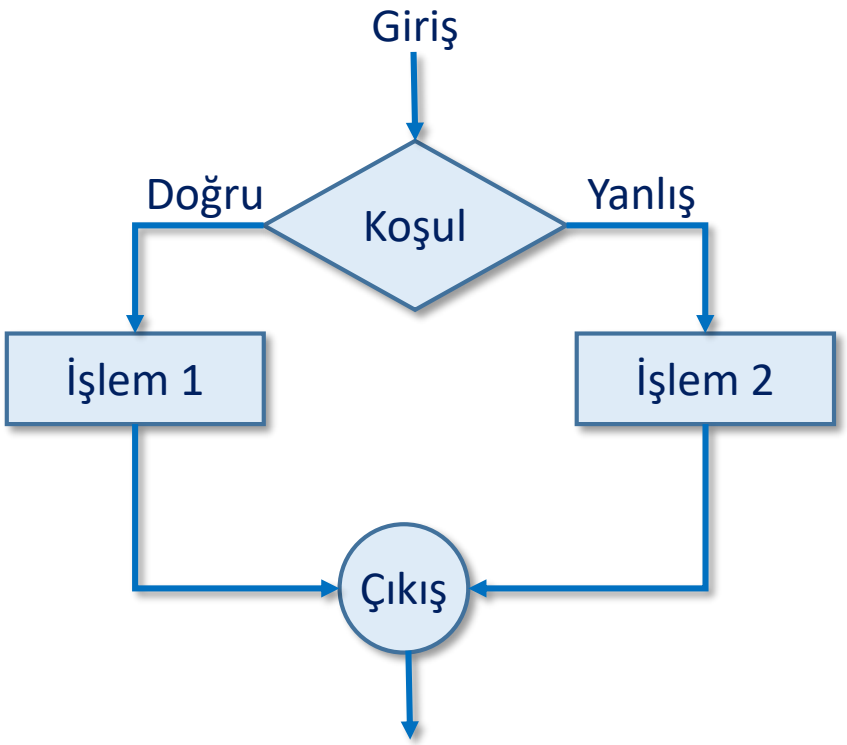
Aynı sayfaya



Ayrı sayfaya

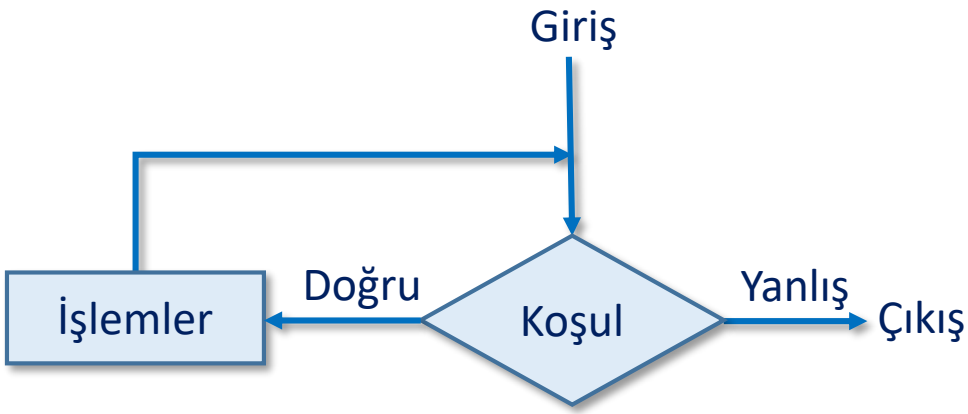
Akış Diyagramı ile Karar / Karşılaştırma Yapısı

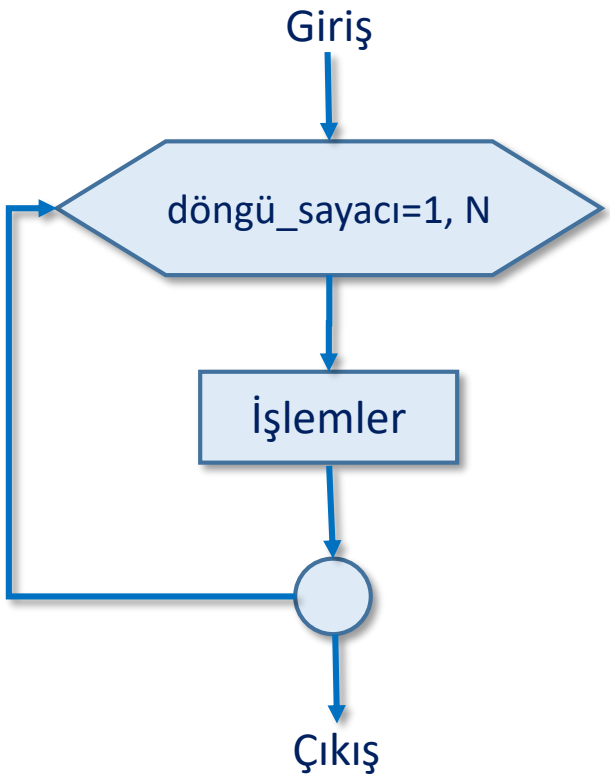
- Algoritma tasarımındaki **EĞER** yapısının gerçekleştirilmesi



Akış Diyagramı ile Tekrar (Döngü) Yapısı

- Algoritma tasarımındaki **Döngü** yapısının gerçekleştirilmesi



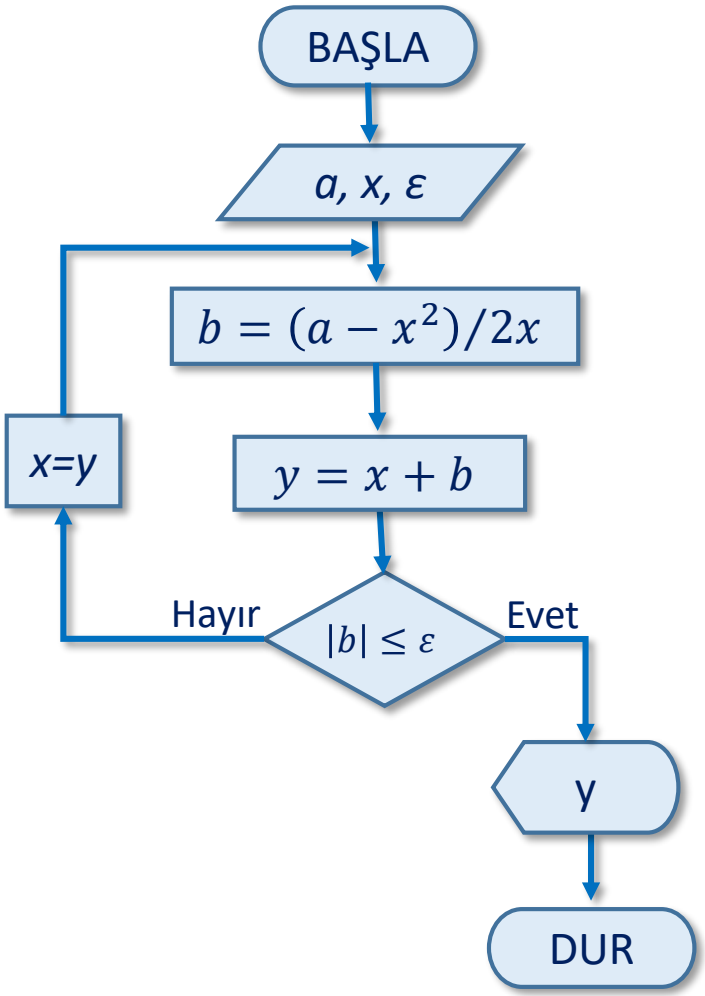


Akış Diyagramı Örnekleri - 1

- **Problem:** Klavyeden okunan bir reel sayının karekökünü bulup sonucu ekrana yazan programın akış diyagramını çiziniz.

- Görüldüğü gibi akış diyagramlarında işlem sırası oklarla gösterildiği için deyimlerin (işlemlerin) A1, A2, vb. etiketlendirilmesine gerek yoktur. Fakat istenirse bu etiketler kutuların dışında sol üst köşeye yazılabilir.

- **Not:** Bu örneğin algoritma tasarımı önceki slaytlarda verilmektedir.



Akış Diyagramı Örnekleri - 2

- **Problem:** 1'den N'ye kadar olan sayıların toplamını hesaplayan programın akış diyagramını çizelim.

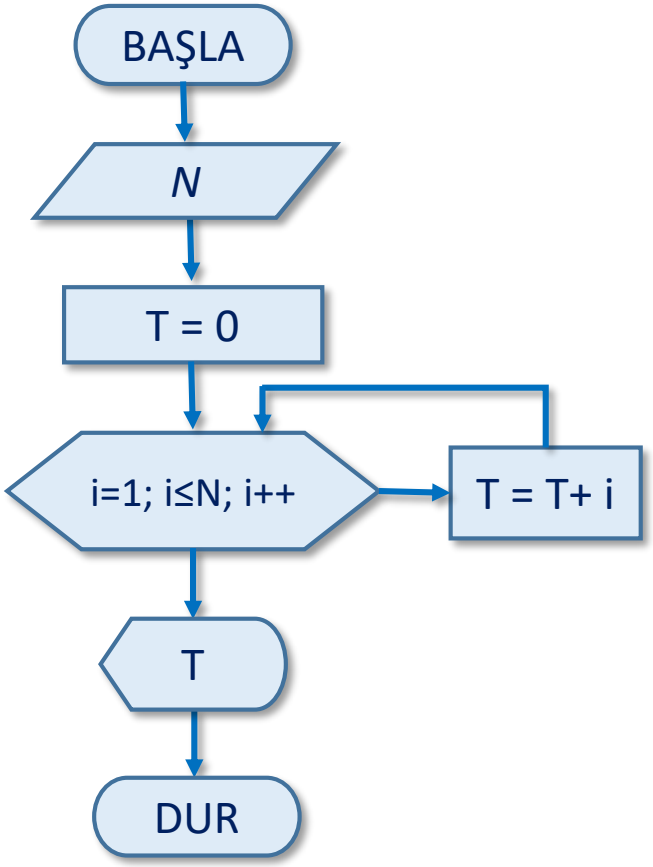
- 1'den N'ye kadar, N adet sayı vardır. Birer artan döngü içinde sayıları toplayabiliriz. Döngü artışını kontrol edeceğimiz değişken i olsun. Toplam değerini de T değişkeni ile ifade edelim. Döngü değişkeni i, 1'den başlayacak ve birer artarak N ye ulaşacak. T başlangıçta 0 ile başlayacak ve döngü içerisinde 1'den N'ye değişen i değeri ilave edilecek.

Akış Diyagramı Örnekleri – 2…

- **Problemin Algoritması:**
  - Algoritma sözde kodlar ile ifade edildiğinde aşağıdaki şekilde yazılabilir:

Adım 1: Başla
Adım 2: Oku (N)
Adım 3: T = 0, i = 0
Adım 4: T = T + i
Adım 5: i = i + 1
Adım 6: Eğer i≤N ise Adım 4'ye git
Adım 7: Yaz (T)
Adım 8: Dur

Akış Diyagramı Örnekleri – 2…



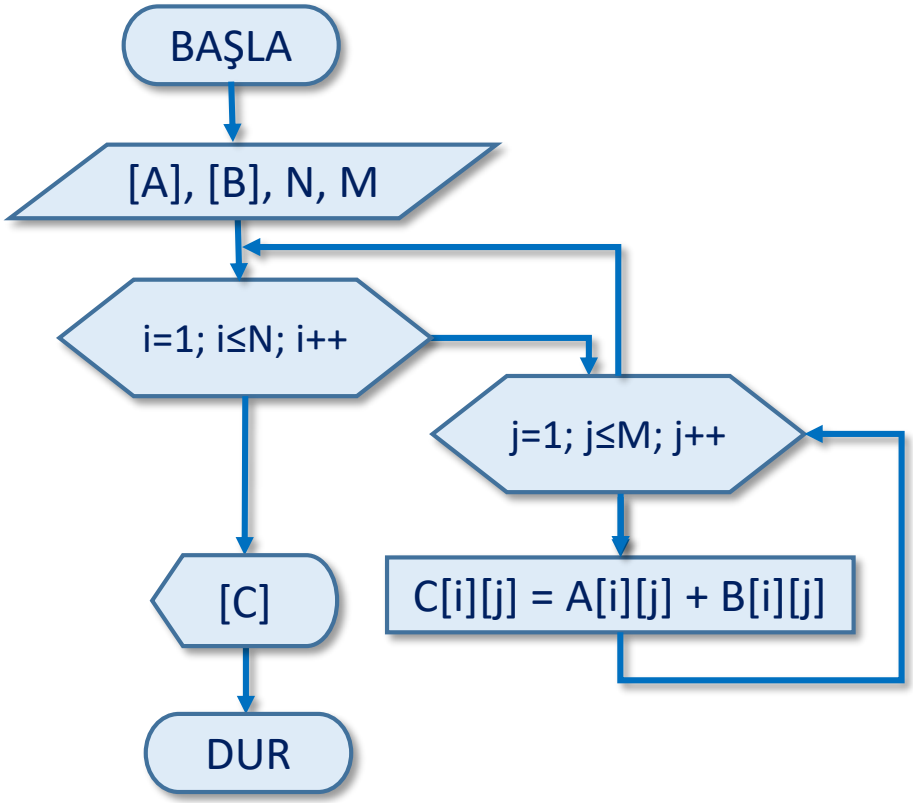
i	N	işlem	T
-	10	-	-
-	10	T=0	0
1	10	T=0+1	1
2	10	T=1+2	3
3	10	T=3+3	6
4	10	T=6+4	10
5	10	T=10+5	15
6	10	T=15+6	21
7	10	T=21+7	28
8	10	T=28+8	36
9	10	T=36+9	45
10	10	T=45+10	55

Akış Diyagramı Örnekleri – 3

- **Problem:** NxM boyutlu iki matrisin toplamını hesaplayan bir algoritmanın akış diyagramını çizelim.

- Matrislerin elemanlarını ifade eden indisleri i ve j ile gösterelim (i=1,...,N, j=1,...,M). Bu durumda A matrisinin her bir elemanı matematiksel olarak $A_{i,j}$ veya programlama açısından A[i][j] şeklinde ifade edilebilir.
- Elemanları toplanacak matrisler A ve B matrisleri ve toplam sonucunda oluşacak matris ise C matrisi olsun.
- Bilindiği gibi matris toplamında birinci matrisin [i][j] indeksli elemanı ile ikinci matrisin [i][j] indeksli elemanı karşılıklı olarak toplanarak toplam matrisin [i][j] indeksli elemanını elde edilir.

Akış Diyagramı Örnekleri – 3…



$$[A]=\begin{bmatrix}1 & 3\\ 1 & 2\end{bmatrix} \quad [B]=\begin{bmatrix}2 & 1\\ 3 & 0\end{bmatrix}$$

matrisleri ve N=M=2 değerleri için

i	j	işlem	C[i][j]
1	1	C[1][1]=1+2	3
1	2	C[1][2]=3+1	4
2	1	C[2][1]=1+3	4
2	2	C[2][2]=2+0	2

$$[C]=\begin{bmatrix}3 & 4\\ 4 & 2\end{bmatrix}$$

KAYNAKLAR

- Deitel, C++ How To Program, Prentice Hall
- Prof. Dr. Cemil ÖZ, Programlamaya Giriş Ders Notları