

SAYISAL ANALİZ

Dr. Öğr. ÜYESİ Abdullah SEVİN



SAYISAL ANALİZ

2. Hafta

ALGORİTMA – MATLAB

Algoritma

- ❑ **Algoritma;** verilerin bilgisayara hangi çevre biriminden girileceğinin, problemin nasıl çözüleceğinin, hangi basamaklardan geçirilerek sonuç alınacağını, sonucun nasıl ve nereye yazılacağını sözel olarak ifade edilmesi biçiminde tanımlanabilir.
- ❑ **Örnek :** Verilen iki sayının toplamının bulunmasının algoritması aşağıdaki gibi yazılır:
 - ❑ Adım 1-Başla
 - ❑ Adım 2-Birinci sayıyı oku
 - ❑ Adım 3-İkinci sayıyı oku
 - ❑ Adım 4-İki sayıyı topla
 - ❑ Adım 5-Dur

PROGRAM GELİŞTİRME

- ❑ **Programlama**, herhangi bir problemin bir programlama dili kullanılarak çözülmesi için yazılan mantıksal kod bloklarına verilen addır.
- ❑ Amaç problemin çözümüne uygun şekilde hazırlanan **program kodu** ile problemi çözmeye çalışmaktır. Bu amaç için araç olarak herhangi bir programlama dilini kullanırız.
- ❑ Programlama diline ait hazır komutları kullanarak problemi çözmeye çalışırız. Bu komutlar programlama dilleri arasında farklılık göstermesine rağmen programlama mantığı bütün dillerde aynıdır.

PROGRAM GELİŞTİRME

- ❑ Unutulmamalıdır ki hazırlanan bir program, gerektiğinde başkaları tarafından da kullanılacaktır. Bu nedenle hazırlanan programın mümkün olduğunca hatalardan arındırılmış olması gerekmektedir. Beklenen sonuçları verecek şekilde hazırlanmış olması gerekmektedir.
- ❑ Bir programlama dilinde hazırlanmış bir program çalıştırılırken genellikle şu iki tür hata ile karşılaşılır:
 - ❑ 1. Yazım hataları,
 - ❑ 2. Mantıksal hatalar.

PROGRAM GELİŞTİRME

- ❑ **Yazım Hataları**, programın derlenmesi sırasında ortaya çıkar ve hata düzeltilmedikçe program çalıştırılmaz.
- ❑ **Mantıksal Hatalar**, yazım hataları gibi programın yazımından kaynaklanan hatalar değildir. Bunlar programın çalıştırılması sırasında ortaya çıkar ve programdan istenen sonucun alınamamasına veya yanlış sonuçlar verilmesine neden olur.
- ❑ Programın hazırlanmasında dikkat edilmesi gereken en önemli konu, problemin **iyi anlaşılması**, **iyi analiz** edilmesidir. Unutulmamalıdır ki bilgisayar sadece programcının vermiş olduğu işlemleri yerine getirir.

PROGRAM GELİŞTİRME

❑ Programlamanın (Program Geliştirmenin) genel yapısı sırasıyla şu adımları kapsar:

1. Problemin tanımlanması,

2. Problemin çözümlenmesi,

2.1. Çözüm yolunun belirlenmesi,

2.2. Çözüm yoluna uygun algoritmanın belirlenmesi,

2.3. Algoritmaya uygun akış diyagramının çıkarılması,

2.4. Algoritmayı gerçekleştirecek uygun programlama dilinin seçilmesi

3. Problemin programlama dili komut seti yardımıyla kodlanması,

4. Hazırlanan programın denenmesi ve belgelendirilmesi.

Problemin tanımlanması

- ❑ Bir problemin herhangi bir programlama dilinde kodlanmasına başlanmadan önce problemin tam olarak anlaşılması gerekmektedir. Aksi halde yanlış çözüm kaçınılmazdır.
- ❑ **Problemin Çözülmesi**
- ❑ **Çözüm Yolunun Belirlenmesi**
- ❑ Giriş verilerinden sonuçta elde edilecek verilere nasıl, hangi yolla ulaşılabileceğinin tespiti gerekir.
- ❑ Bu durumun iyi analiz edilmesi gerekir.
- ❑ Problemin matematiksel modeli bu aşamada belirlenir.
- ❑ Hangi tekniğin en uygun olduğuna programcının bilgisi ve tekniği etki eder.

Problemin tanımlanması

- ❑ Örnek:
- ❑ Aranan bir büyüklüğün herhangi bir $\{a\}$ kümesi içerisinde olup olmadığının araştırılması.
- ❑ $\{a\} = \{3, 7, -10, 8, 1, -4, -94, 6, 2, -1, 34, 14, 78, -19, 99\}$ olsun.
- ❑ $x = -12$ elemanının bu küme içinde yer alıp almadığını arayalım.

Problemin tanımlanması

❑ Çözüm Yolları:

1. Verilen x değeri (-12) sırayla $\{a\}$ kümesinin bütün elemanları ile tek tek karşılaştırılarak arama yapılabilir.

2. Önce $\{a\}$ kümesi kendi içerisinde artan sırada (büyükten küçüğe doğru) sıralanır.

$$\{a\} = \{-94, -19, -10, -4, -1, 1, 2, 3, 6, 7, 8, 14, 34, 78, 99\}$$

Daha sonra verilen x değeri (-12) sıralanmış $\{a\}$ kümesi içerisinde baştan bütün elemanlar ile karşılaştırılarak arama yapılabilir. En son karşılaştırılan değer x değerinden büyük ise işlem kesilir.

Problemin tanımlanması

❑ Çözüm Yolları:

3. Önce $\{a\}$ kümesi kendi içerisinde artan sırada (büyükten küçüğe doğru) sıralanır.

$$\{a\} = \{-94, -19, -10, -4, -1, 1, 2, 3, 6, 7, 8, 14, 34, 78, 99\}$$

$\{a\}$ kümesindeki eleman sayısı ikiye tam bölünür ve orta eleman bulunur. x değeri (-12) orta elemanla karşılaştırılır. Eğer orta eleman x değerinden büyük ise x değeri $\{a\}$ kümesinin ilk yarısında olacaktır. İlk yarıdaki eleman sayısı ikiye tam bölünerek 2. orta eleman bulunur. x değeri 2. orta elemanla karşılaştırılır.

Eğer 2. orta eleman x değerinden büyükse x $\{a\}$ kümesinin 2. yarısının ilk bölümünde olacaktır. Bu işlemler tek eleman kalıncaya kadar sürdürülür. Arama sonlandırılır.

$$\{a\}' = \{-94, -19, -10, -4, -1, 1, 2, 3\}$$

$$\{a\}'' = \{-94, -19, -10, -4\}$$

$$\{a\}''' = \{-10, -4\}$$

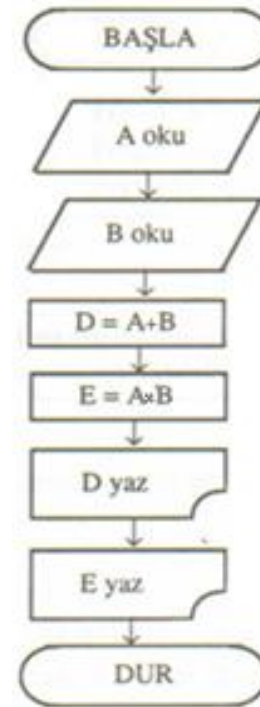
Problemin tanımlanması

- ☐ **Çözüm Yolları:**
- ☐ Yukarıda belirtilen 3 farklı çözüm yolu problemin çözümünü sağlamaktadır.
- ☐ **Bu çözüm yollarından hangisinin seçileceği;**
- ☐ Verinin büyüklüğüne,
- ☐ Amaçlanan işlem hızına,
- ☐ Yapılacak işlemin tekrarlanma sayısına bağlıdır.

Akış Şemaları (Diyagramları)

- ❑ Herhangi bir sorunun çözümü için izlenmesi gerekli olan aritmetik ve mantıksal adımların söz veya yazı ile anlatıldığı algoritmanın, görsel olarak simge ya da sembollerle ifade edilmiş şekline **"akış şemaları" (FLOWCHART)** adı verilir. Örnek;








- Adım 1-Başla
- Adım 2-A'yı oku
- Adım 3-B'yi oku
- Adım 4- $D=A+B$
- Adım 5- $E=A*B$
- Adım 6-D'yi yaz
- Adım 7-E'yi yaz
- Adım 8-Dur



Akış Şemaları (Diyagramları)

- ❑ Akış şemalarının hazırlanmasında aşağıda yer alan simgeler kullanılır.
- ❑ Akış şemaları içerik ve biçimlerine göre genel olarak üç grupta sınıflandırılabilirler.

- ❑ Doğrusal Akış Şemaları,
- ❑ Mantıksal Akış Şemaları,
- ❑ Döngüsel Akış Şemaları,

	Algoritmanın başladığını ya da sona erdiğini belirtmek için kullanılır.
	Klavye aracılığı ile giriş ya da okuma yapılacağını gösterir.
	Yazıcı(printer) aracılığı ile çıkış yapılacağını gösterir.
	Kart okuyucu aracılığıyla giriş yapılacağını gösterir.
	Araç belirtmeden giriş ya da çıkış yapılacağını gösterir.
	Hesaplama ya da değerlerin değişkenlere aktarımını gösterir.
	Aritmetik ve mantıksal ifadeler için karar verme ya da karşılaştırma durumunu gösterir.

Doğrusal Akış Şemaları

❑ İş akışları, giriş, hesaplama, çıkış biçiminde olan akış şemaları bu grup kapsamına girer.

❑ Akış Şeması

A: Birinci sayı B: İkinci sayı

D: İki sayının toplamını ($A+B$)

E: İki sayının bölümünü ($A \times B$)

❑ Algoritma

Adım 1-Başla

Adım 2-A'yı oku

Adım 3-B'yi oku

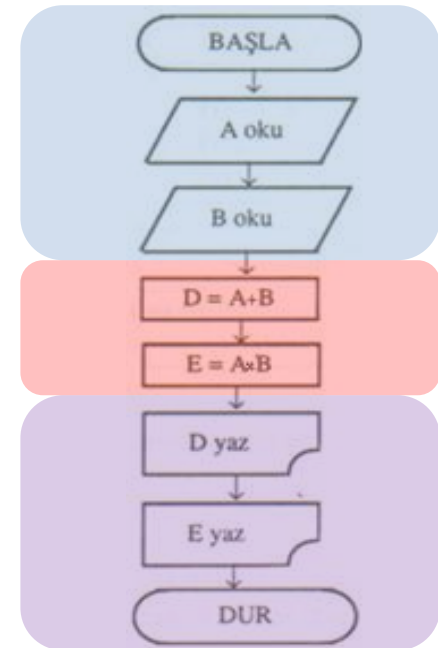
Adım 4- $D=A+B$

Adım 5- $E=A \times B$

Adım 6-D'yi yaz

Adım 7-E'yi yaz

Adım 8-Dur



Mantıksal Akış Şemaları

❑ Mantıksal kararları içeren akış diyagramlarıdır

❑ Algoritma

Adım 1-Başla

Adım 2-X,Y'yi oku

Adım 3- $X > Y$ ise Adım 6'e git

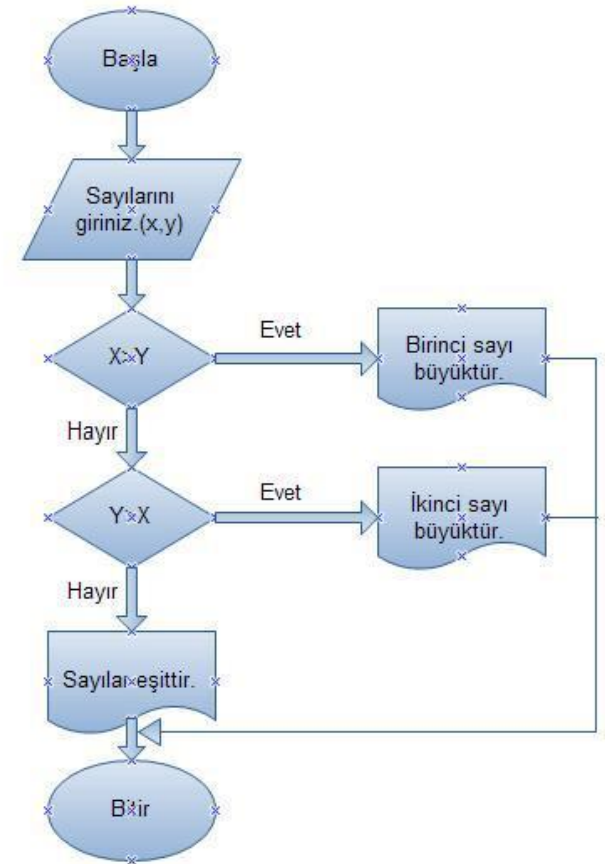
Adım 4- $X < Y$ ise Adım 7'ya git

Adım 5- 'X ve Y eşit' mesajını yaz Adım 8'e git

Adım 6- 'X BÜYÜKTÜR' mesajını yaz Adım 8'e git

Adım 7- 'Y BÜYÜKTÜR' mesajını yaz

Adım 8-Dur



Döngüsel Akış Şemaları

❑ Akış sürecinde yer alan herhangi bir adım ya da aşamanın birden fazla kullanıldığı akış diyagramlarına denir.

❑ Örnek :

N sayısını ekrandan okutarak faktöriyelini hesaplayıp yazan programın algoritma ve akış diyagramını oluşturalım.

❑ Değişkenler

NFAK=N faktöriyel (N!) değerini,

OGRSAY=1'den N'e kadar sayıları gösterecek,

NFAK=1*2*.....*N

❑ Algoritma

Adım 1-Başla

Adım 2-N'i ekrandan oku

Adım 3-NFAK=1

Adım 4-OGRSAY=1

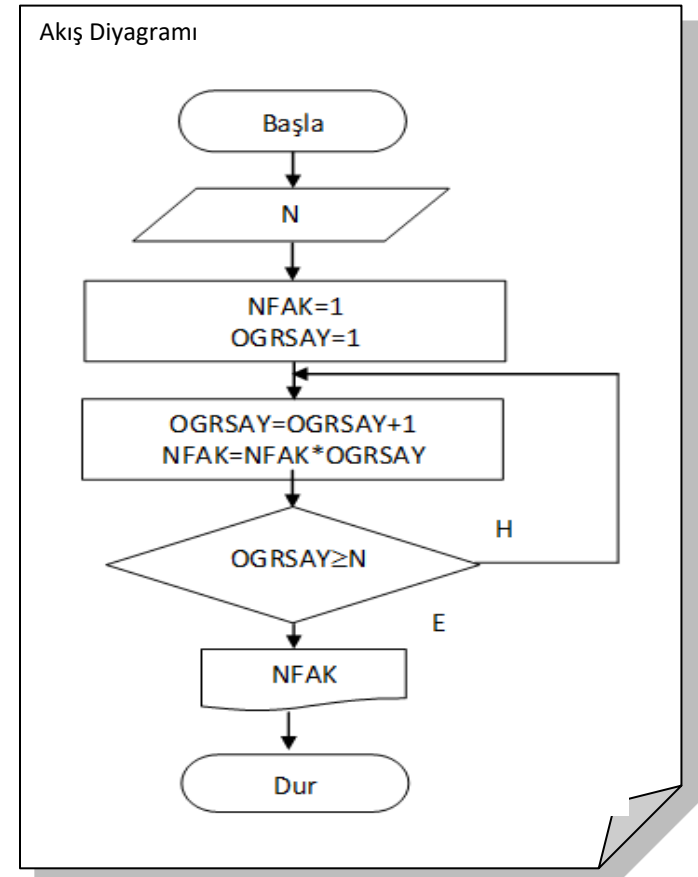
Adım 5-OGRSAY=OGRSAY+1

Adım 6-NFAK=NFAK*OGRSAY

Adım 7-Eğer OGRSAY < N ise adım 5'e git

Adım 8-NFAK yaz

Adım 9-Dur

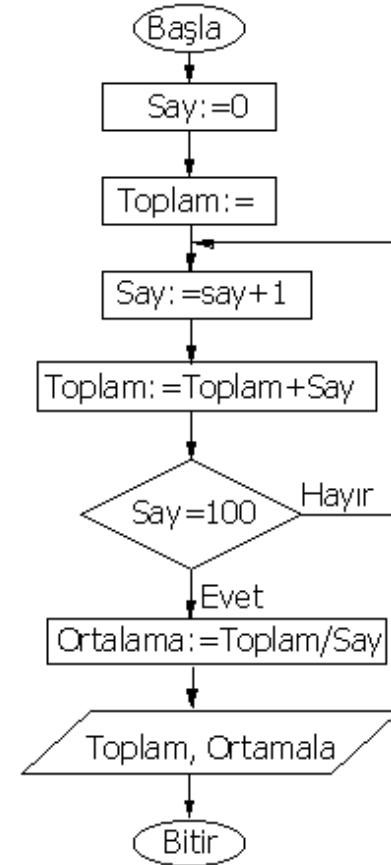


Uygulama 1

❑ 1'den 100'e kadar olan sayıların toplamalarını ve ortalamalarını veren programın akış diyagramını çiziniz. .

❑ Algoritma

1. Adım: Say=0
2. Adım: Toplam = 0 olsun.
3. Adım: Say++
4. Adım: Toplam = Toplam+Say;
5. Adım: Eğer Say !=100 ise Git Adım 3
6. Adım: Ort = Toplam/Say
7. Adım: Yaz toplam, Ort
8. Adım: DUR.



Uygulama 2

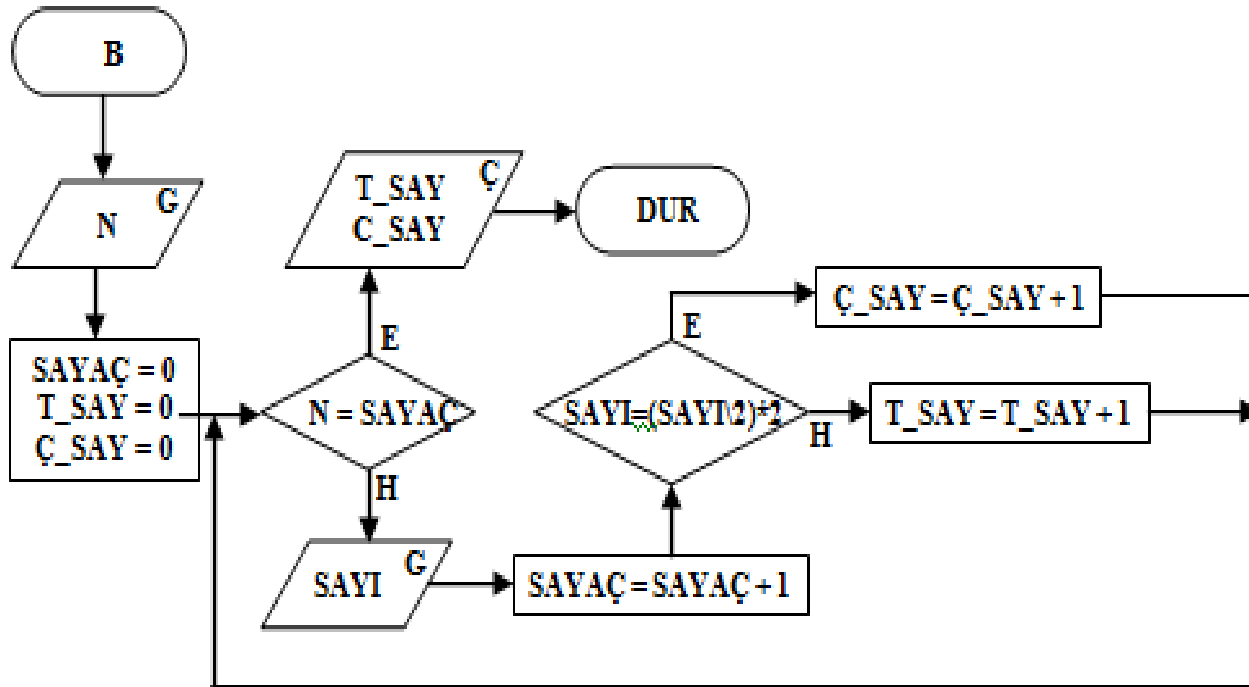
❑ İsteğe bağlı sayıda girilen sayıların içinden tek ve çift olanların sayısını bulduran algoritmayı yazarak ve akış diyagramını çiziniz.

❑ Algoritma :

1. Adım: N sayısını al.
2. Adım: SAYAÇ = 0 olsun.
3. Adım: T_SAY = 0 olsun.
4. Adım: Ç_SAY = 0 olsun.
5. Adım: Eğer $N = \text{SAYAÇ}$ ise 11. Adım' a git.
6. Adım: Bir SAYI al.
7. Adım: $\text{SAYAÇ} = \text{SAYAÇ} + 1$
8. Adım: Eğer $\text{SAYI} = (\text{SAYI} \setminus 2) * 2$ ise $\text{Ç_SAY} = \text{Ç_SAY} + 1$ ve Adım 5'e git
9. Adım: $\text{T_SAY} = \text{T_SAY} + 1$
10. Adım: 5. Adım' a git.
11. Adım: T_SAY yaz.
12. Adım: Ç_SAY yaz.
13. Adım: DUR.

Uygulama 2

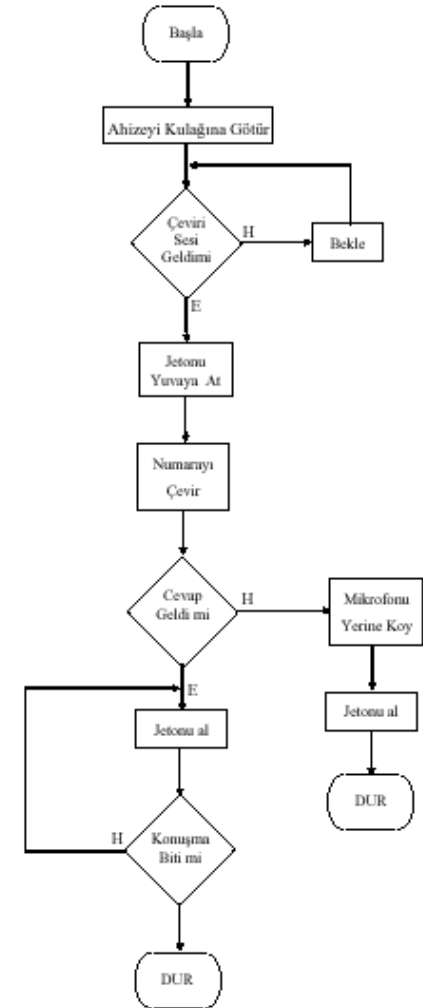
- İsteğe bağlı sayıda girilen sayıların içinden tek ve çift olanların sayısını bulduran algoritmayı yazarak ve akış diyagramını çiziniz.
- Akış Şeması :



Uygulama 3

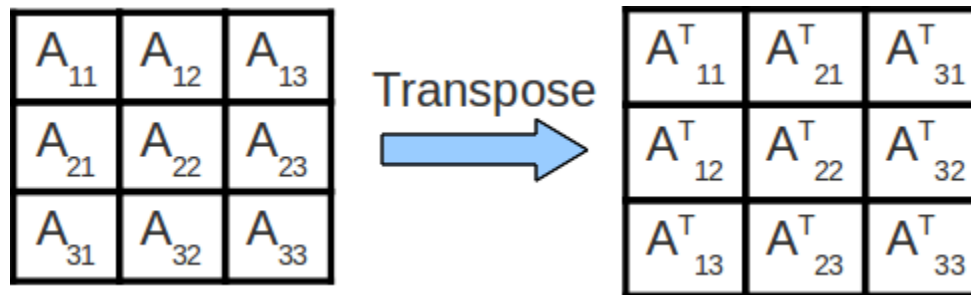
☐ Telefonla Görüşmenin Algoritması

1. Ahizeyi kulağınıza götürünüz
2. Çevir sesinin gelmesini bekle
3. Ses gelmişse (E) jetonu ilgili yuvaya at
4. Ses gelmemişse bekle
5. Ses gelirken jeton yerine yerleşince numarayı çevir
6. Cevap geldi mi?
7. Hayır ise ahizeyi yerine koy
8. Jetonu iade çıkısından al
9. Cevap gelmişse konuş
10. Konuşma bitti mi?
11. Evet ise Ahizeyi yerine koy
12. Hayır ise konuşmaya devam et.



Uygulama 4-5

- ❑ Uygulama 4
- ❑ $m \times n$ tipinde klavyeden girilen matrisin transpozunu ekrana yazan programın akış şemasını çiziniz.



- ❑ Uygulama 5
- ❑ n boyutlu üçgensel (alt, üst, köşegen) matrisin akış diyagramını çiziniz.

Upper triangular matrix: U

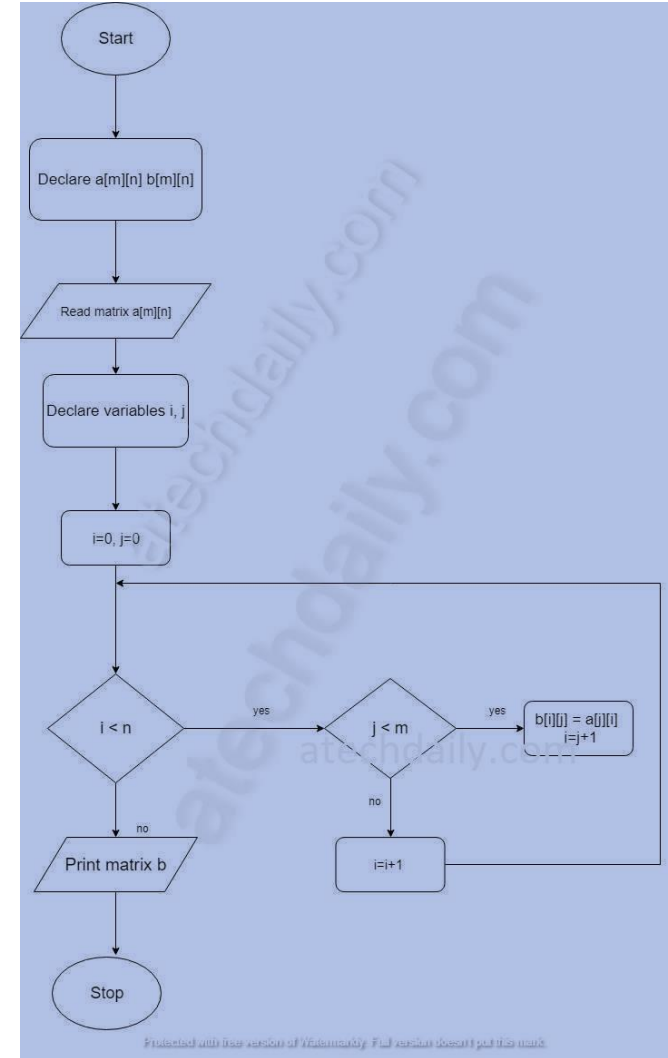
$$\begin{bmatrix} 1 & 1/2 & 3 & 0 \\ 0 & 5 & 0 & 1 \\ 0 & 0 & 4 & -2 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Lower triangular matrix: L

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 3 & 0 \\ 1 & -2 & 0 \end{bmatrix}$$

Uygulama 4-5

- ❑ Uygulama 4
- ❑ $m \times n$ tipinde klavyeden girilen matrisin transpozunu ekrana yazan programın akış şemasını çiziniz.



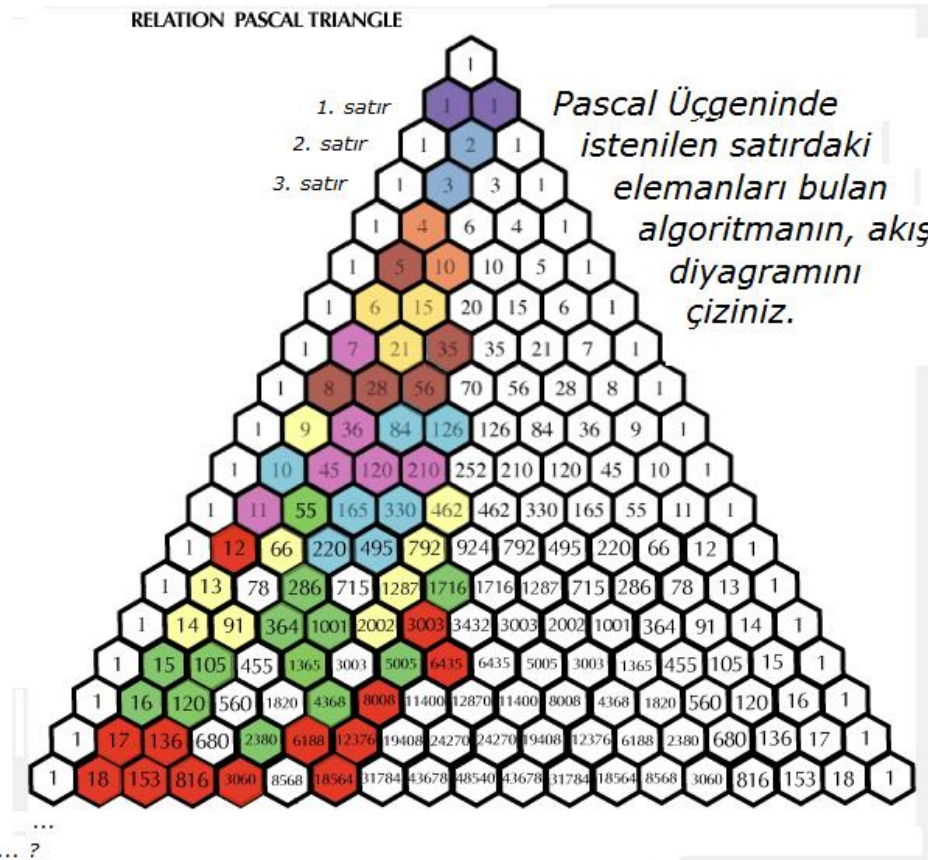
Uygulama 4-5

- ❑ Uygulama 5
- ❑ n boyutlu üçgensel (alt,üst,köşegen) matrisin akış diyagramını çiziniz.

```
void lower(int matrix[3][3], int row, int col)
{
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            if (i < j)
            {
                cout << "0" << " ";
            }
            else
            {
                cout << matrix[i][j] << " ";
            }
        }
        cout << endl;
    }
}
```


Uygulama Soru

- ❑ Pascal üçgeni, binom açılımındaki katsayıları bulmaya yarar. Pascal'ın bu üçgeni, olasılıklar kuramında da ustalıkla kullanılır. Bu üçgen, biyolojideki uygulamalar, matematik, istatistik ve pek çok modern fizik konularında uygulama alanı bulur..



KAYNAKLAR

❖ Temel Kaynaklar

- Ders Notları – Sunular

❖ Diğer Kaynaklar

- Steven C. Chapra, Raymond P. Canale (Çev. H. Heperkan ve U. Kesgin), “*Yazılım ve Programlama Uygulamalarıyla Mühendisler İçin Sayısal Yöntemler*”, Literatür Yayıncılık.
- Serhat YILMAZ, “*Bilgisayar İle Sayısal Çözümleme*”, Kocaeli Üniv. Yayınları, No:168, Kocaeli, 2005.
- İlyas ÇANKAYA, Devrim AKGÜN, Sezgin KAÇAR “*Mühendislik Uygulamaları İçin MATLAB*”,Seçkin Yayıncılık
- Mehmet Bakioglu, “*Sayısal Analiz*”, Birsen Yayınevi, 2004.
- Yüksel YURTAY, Sayısal Analiz Ders Notları, Sakarya Üniversitesi
- Fahri VATANSEVER, “*İleri Programlama Uygulamaları*”,Seçkin Yayıncılık
- İrfan Karagöz, “*Sayısal Analiz ve Mühendislik Uygulamaları*”, VİPAŞ Yayınevi, 2001.