

ALGORİTMA 1 DERS NOTLARI

Yrd. Doç. Dr. Pakize ERDOĞMUŞ

01.01.2011

Duzce Üniversitesi

BİLGİSAYAR MÜHENDİSLİĞİ BASLA

x. DERSINI AL

Gecti
n mi?

Hayır

Bu dersi geçemediğin için
x+1. Ve x+2. Dersi
alamayacaksın.

X+1. Dersi al

Evet

Gecti
n mi?

Hayır

Bu dersi geçemediğin için
x+2. dersi alamayacaksın.

.....

MEZUN OLDUN

GİRİŞ

Bu ders notu 2010-2011 bahar yarıyılından itibaren yürüttüğüm C++ ile Algoritma ve Programlama I dersi için hazırlanmıştır.

Konular akış sırasına göre işlenmektedir. Ancak bazen örnek uygulamalar için bazı ön bilgiler(C++ hazır fonksiyonu veya programlama bilgisi) gerekmektedir. Bu ön bilgiler gölgeli tablolar içinde verilmiştir.

Ders notu hazırlanırken yurt içinden ve yurt dışından çok sayıda ders notu ve C++ ders kitabı incelenmiştir.

Bilgisayar Programlama, ders ve ders dışı etkinliklerle gelişecektir. Bu sebeple bu notların sadece temel kaynak olarak verildiği unutulmamalıdır. Ders notunun bir başka faydası her hafta ne öğrenileceğini önceden bilmenizdir. Dersten önce notu inceleyip, o hafta ki konu ile ilgili araştırma yapmanızı, örnek programlar yapmanızı tavsiye ederim.

Bu dersi aldıktan sonra;

- Herhangi bir programın algoritmasını çizebilirsiniz.
- C++ programlama dilinin temel prensiplerini bilirsiniz.
- C++ ile döngüler içeren programlar yazabilirsiniz.
- Her türlü matematiksel probleme çözüm getirecek programlar yazabilirsiniz.

Neler yapamazsınız:

-C++, programlama mantığının kavranması için verildiğinden görsel arayüz anlatılmamaktadır. Bu sebeple görselliği iyi programları henüz yazamazsınız.

-Veri tabanı veya dosyalama işlemleri olan programlar yazamazsınız.



DERS BİLGİ FORMU

ENSTİTÜ/FAKÜLTE/YÜKSEKOKUL ve PROGRAM:

DÜZCE ÜNİVERSİTESİ, BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERS BİLGİLERİ

Adı	Kodu	Dili	Türü Zorunlu/ Seçmeli	Yarıyılı	T+U Saati	Kredisi	AKTS
Algoritmalar ve Programlama 2	BM 102	Turkish	Z	2	3	3	5

Ön Koşul Dersleri	BM 101
-------------------	---------------

Ders Sorumluları	Yrd. Doç. Dr. Pakize ERDOĞMUŞ
Ders Sorumlu Yardımcıları	Arş. Gör. Sultan Zavrak, Arş. Gör. Zehra Karapınar Şentürk, Arş. Gör. Arafat Şentürk

Dersin Amacı	Öğrencilere, C++ dilini kullanarak yapısal programa dilleri hakkında bilgi, beceri ve deneyim kazandırmak. Yapısal programlama ve Nesneye Yönelimli Programlama dilleri ile programlama yazma yeteneği kazandırmak.
--------------	---

Dersin Öğrenme Çıktıları	<p>Dersi tamaldıktan sonra öğrenci:</p> <p>Bilgi:</p> <ul style="list-style-type: none"> • Nesneye Dayalı Program geliştirmeyi bilir. • C++ programlama dilinin yapısını bilir. <p>Yetkinlik</p> <ul style="list-style-type: none"> • Küçük ve orta ölçekli C++ programları geliştirir.
--------------------------	--

DERS PLANI			
Hafta	Ön Hazırlık	Konular/Uygulamalar	Metot
1		Ders tanıtım, Algoritma Giriş(1. Bölüm)	Klasik anlatım,sunum, Program uygulaması
2		Programlamaya Giriş(2. Bölüm)	"
3		Algoritma hazırlama ve analizi(3. Bölüm)	"
4		Akiş Diyagramları(4. Bölüm)	"
5		Akiş Diyagramları(4. Bölüm devam)	"
6		Veri Tipleri(5. Bölüm)	"
7		Programlama Dilleri ve C++ ile Programlama(6.1 Bölüm)	"
8		C++ Programlamada Yazım kuralları(6.2)	"
9		C++ Programlamada döngü(7. Bölüm)	"
10		Vize Haftası	"
11		C++ Programlamada Kontrol ve Döngü İfadeleri Devam(7. Bölüm devam)	"
12		C++’da tek boyutlu diziler	"
13		C++’da iki boyutlu diziler	"
14		Genel Tekrar, Uygulamalar	"

KAYNAKLAR	
Ders Kitabı veya Notu	C++ Ders Notum ve verdığım soft copy kaynaklar
Diğer Kaynaklar	<ul style="list-style-type: none"> • Algoritma Tasarımı ve Program Geliştirme, Fahri Vatansever, Seçkin Yayınevi. • Bilgisayar Yazılımında Veri Yapıları ve Algoritma, M. Ümit Karakaş, Seçkin Y.E. • Algoritma Tasarlama ve Programlamaya Giriş, Selami Eryılmaz, Seçkin Y.E. • Problem Solving, Abstraction, and Design Using C++ (<i>Frank L. Friedman and Elliot B. Koffman</i>) Addison Wesley Inc. ISBN:0-201-52649-2 • C ve C++ (C: How to Program) Yazarı: Harvey M. Deitel & Paul J. Deitel Çevirmen: Metin Zavrak,Ekrem Aksoy, H. Nihal Karaca, Sistem Yayıncılık. ISBN: 975322307-2 • C/C++ Programcının Rehberi (Chris H. Pappas William H. Murray Cev: Kadir Ertürk) Sistem Yayıncılık
Dersin Kuralları	<p>Dersin planının iyice inceleyiniz.</p> <ol style="list-style-type: none"> 1. Quiz tarihleri ders içerisinde bir hafta öncesinde ilan edilir. 2. Yoklama dersin ilk 15 dakikasında alınır. 15 dakikadan sonra gelenler derse girebilir ancak yoklamaya imza atamaz. Aktif olarak işlenen ve yoklama alınan dersler den %100 devam edenler +5 puan kazanır. 3. Dersin yazılı ödevlerini sınıf temsilcisi <u>imza karşılığında toplar</u> ve teslim eder. Programlar ise o ödev için açılan klasöre atılır. Hocanın belirttiği tarihten sonra klasöre erişilemez. 4. Programlama ve yazılı ödevlerde aynı olan ödevlere puan verilmez. 5. <u>BU DERSİ YÜKSELTME İÇİN ALANLAR(%40 Vize+%60 FINAL) NOTU İLE DEĞERLENDİRILECEKTİR:</u>
DEĞERLENDİRME SİSTEMİ	

Etkinlik Türleri	Katkı Yüzdesi
Ara Sınav	%30
Kısa Sınav	%20
Ödev, Proje	%10
Yarıyıl Sonu Sınavı	%40
Toplam	%100

DERSİN PROGRAM ÇİKTILARINA KATKISI						
No	Program Çıktıları	Katkı Düzeyi				
		1	2	3	4	5
1	Mühendislik eğitimi için gerekli matematik altyapısına sahip olarak, analitik düşünme yeteneği kazandırmak				x	
2	Problemleri matematiksel formüllerle ifade edebilme ve sayısal analiz yapabilmek					x
3	Bilişim teknolojilerini takip ederek kazanılan bilgiyi güncel tutabilmek					x
4	Güncel web teknolojilerini etkin şekilde kullanarak mevcut yazılımları web teknolojileriyle birleştirebilmek	x				
5	İleri düzey programlar yazabilecek algoritmalar geliştirebilme yeteneğine sahip olmak					x
6	Verileri analiz ederek sonuç elde edebilme yeteneğine sahip olmak					x
7	Bilgi teknolojilerini kullanarak diğer disiplinlerde oluşan problemlere çözüm üretabilmek				x	
8	Problem çözümünde gerekli disiplinleri takip ederek uygun çözüme erişebilmek		x			
9	Mühendislik bilgi birikimini toplumun sosyal problemlerini çözmede kullanabilme becerisine sahip olabilmek			x		
10	Sayısal sistemlerin yapısını anlayabilme, yeni sistem tasarlayabilme, sayısal sistem tasarımcılık için gerekli donanım altyapısına sahip olma, bilgisayar ağlarını planlayabilmek ve uygulayabilmek için gerekli alt yapıya sahip olabilmek					x

AKTS / İŞ YÜKÜ TABLOSU		İş Yükü (Saat)
Ders İçi	Ders Saati (14 x Haftalık Ders Saati)	14x3=42
Ders Dışı	Ödev	14X3=42
	Araştırma	14
	Ön Hazırlık, Pekiştirme Çalışmaları	14
	Diğer Faaliyetler	14
Sınavlar	Ara Sınav (Ara Sınav Sayısı x Ara Sınav Süresi)	5
	Yarıyıl Sonu Sınavı	5

Toplam İş Yükü	136
Toplam İş Yükü / 30 (s)	136/30
Dersin AKTS Kredisi	5

Doç. Dr. Pakize ERDOĞMUŞ

1. Algoritma ve Programlama

Genel anlamda algoritma çözümlenmesi gereken bir problemin işlem adımlarının çıkarılmasıdır diyebiliriz. Bilgisayar Mühendisliğinde ise algoritma bir bilgisayar programının çözümünü içeren işlem adımlarının simgesel(akış şeması) veya yazı ile ifadesidir.

Algoritma yaklaşımı, ilk olarak 9. yüzyılda yaşamış Türk-İslam matematikçi ve astronomu *Harzemli Mehmet*'in ikinci derece denklemlerin çözümü için geliştirdiği bir yöntemdir[1]. Algoritma kelimesi de aritmetik problemleri çözme kuralları anlamında Harizmi'nin ismine atıfta bulunularak "algorizm" kelimesinden türetilmiştir.

Bilgisayar mühendisliğinde neden algoritmaya ihtiyaç duyulur? Bir bilgisayar programı birçok görev için yazılır ve çok satıldan oluşur. Algoritma yazmadan yani program üzerinde hiç düşünmeden ve bir taslak çıkarmadan kodlamaya geçilirse birçok hata ile karşılaşılır. Bu sebeple önce programdan istenenler nedir ve işlem adımları nelerdir diye düşünülerek algoritma yazılır. Algoritma bilgisayar programının krokisi gibi düşünülebilir. Algoritma günlük dilde yazılabileceği gibi algoritma için geliştirilmiş diller ile de yazılabilir. Acaba algoritma bir yazılımın geliştirilmesi aşamasında nerede yer alıyor bunu görebilmek için, bir yazılım(software) programının geliştirilme asamaları kısaca aşağıda verilmiştir.

1. Analiz: Önce programdan neler bekleniyor. Giriş verileri nedir? Çıkışta neler isteniyor bunlar belirlenir.
2. **Algoritma:** Bu programın yapacağı işlemlerin adım adım tanımlandığı işlemler sırasıdır. Bu işlemler yazılı olarak(pseudo-kod) ile ifade edilebileceği gibi çeşitli şekiller ile de ifade edilebilirler. İşte algoritmanın şekiller ile ifade edilmesine akış şeması adı verilir.
3. Kodlama: Akış şeması veya algoritması çıkarılan programın bir programlama dili ile ifade edilmesidir.
4. Derleme: Program kodlarının makine diline dönüştürülmesi işlemidir.
5. Test(Debugging): Programın doğru çalışıp çalışmadığı kontrol edilir. Hatalardan arındırılır.
6. Dökümantasyon: Bu programı ilk defa kullanacak kişiler için belge hazırlanır.

Göründüğü gibi program geliştirme aşamalarının en önemli algoritma geliştirmedir. Program kodlama algoritmanın bilgisayara aktarılmasından başka bir şey değildir. Algoritma mantığı oturmuşsa program geliştirmek kolay olacaktır.

Program geliştirildikten sonra derlenir. Bilgisayarın programımızı çalıştırabilmesi için **executable program** haline getirilir. Yani yazılan kodların işletilebilmesi için makine diline çevrilir.

Makine dili, bir bilgisayarın anlayabileceği tek dildir. Makine dili ile yazılan programlar yalnızca 0 ve 1'lerden oluşur. Bu dille program yazabilmek için CPU'yu iyi bilmek gereklidir. Makine dilinde programlama çok karmaşık olduğundan makine dili kodları simgesel (assembly) dili ile ifade edilir. **Simgesel (assembly) dillerde**, 0 ve 1'ler yerine bazı sözcükler ve simgeler kullanılır.

Günümüzde bir bilgisayarlı sistemin tamamını anlamak mümkün değildir. Ama en basit bir mikroişlemci(bilgisayarın beyni) sistemin çalışma mantığı oldukça kolaydır.

Bir mikroişlemci ve bir bellek bir mikroişlemci sistemi oluşturur. Bu sistemin çalışması ise **Fetch-Execute** ikilisi ile açıklanabilir.

Ama önce bilgisayar bilimlerinde önemli bazı kavramları anlamaya çalışalım.

Veri(Data): Bilgisayara depolayacağımız veya üzerinde işlem yapılacak her türlü bilgiye veri denilir. Temel olarak iki tip veri vardır:

Sayısal veri: Rakamlardan oluşan bilgilerdir. $0,566,1 \times 10^{34}$ sayısal veriye örnektir.

Alfasayısal veri: Sayısal olmayan her türlü veriye alfasayısal veri adı verilir. Veri tabanına kaydedilecek isim, telefon numarası, kimlik bilgileri alfa sayısal verilerdir. Bu verilerin hepsi bir bilgisayar için alfasayısal veri olup, sayısal veriden daha fazla yer kaplar. İşte ister sayısal, ister alfasayısal olsun verilerin uzunluğunu ölçen en küçük birim bit'tir.

Bit: Bilgisayarda kullanılan en küçük bilgi saklama birimidir. 0 veya 1 olabilir.(Elektriğin olması(1) ve olmaması(0)). Verilerin uzunluğu bit sayısı ile ölçülür. Dolayısıyla bilgisayarlarda binary(ikili) sayı sistemi kullanılır.

Byte: 8 bitten oluşur. 00000000-11111111 arası 256 farklı byte mevcuttur. Bu şekilde 8 bitlik bilgi ile bile 256 karakter kodlanabilir. ASCII(American Standard Codes for International) kod tablosu klavyede bulunan karakter ve tuşlar için böyle bir kodlama yapmıştır. Örneğin A'nın ikili kodu $65=(0100\ 0001)_2$ (Not: Basamaklar 1'ler, 2'ler, 4'ler, 8'ler, ..., 128'lerdir.), a'nın ikili kodu ise $97=(0110\ 0001)_2$ dir.

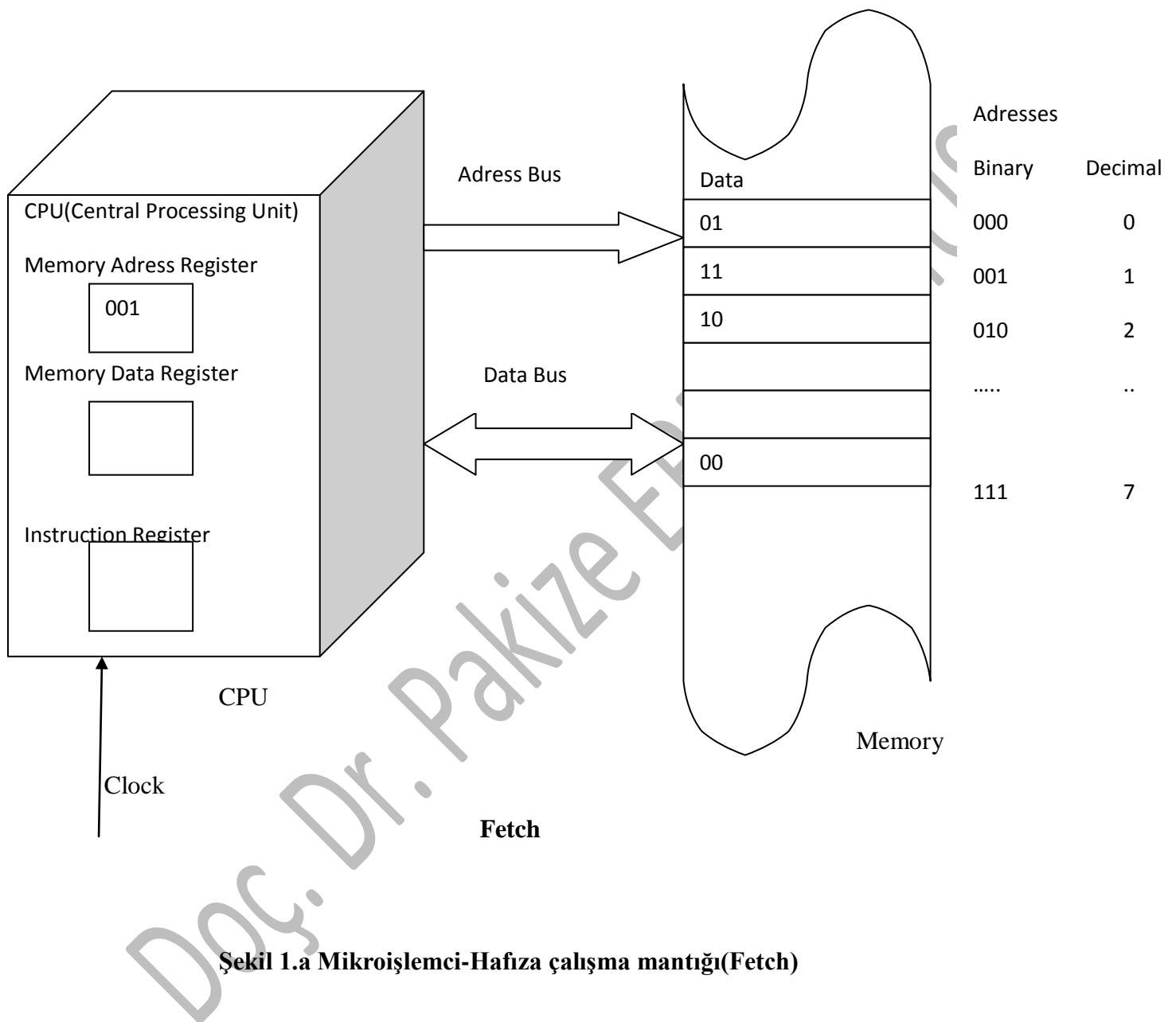
ASCII kod tablosunda 48-57 arası rakamlar, 65-90 arası büyük harfler ve 97-122 arası küçük harfler mevcuttur.

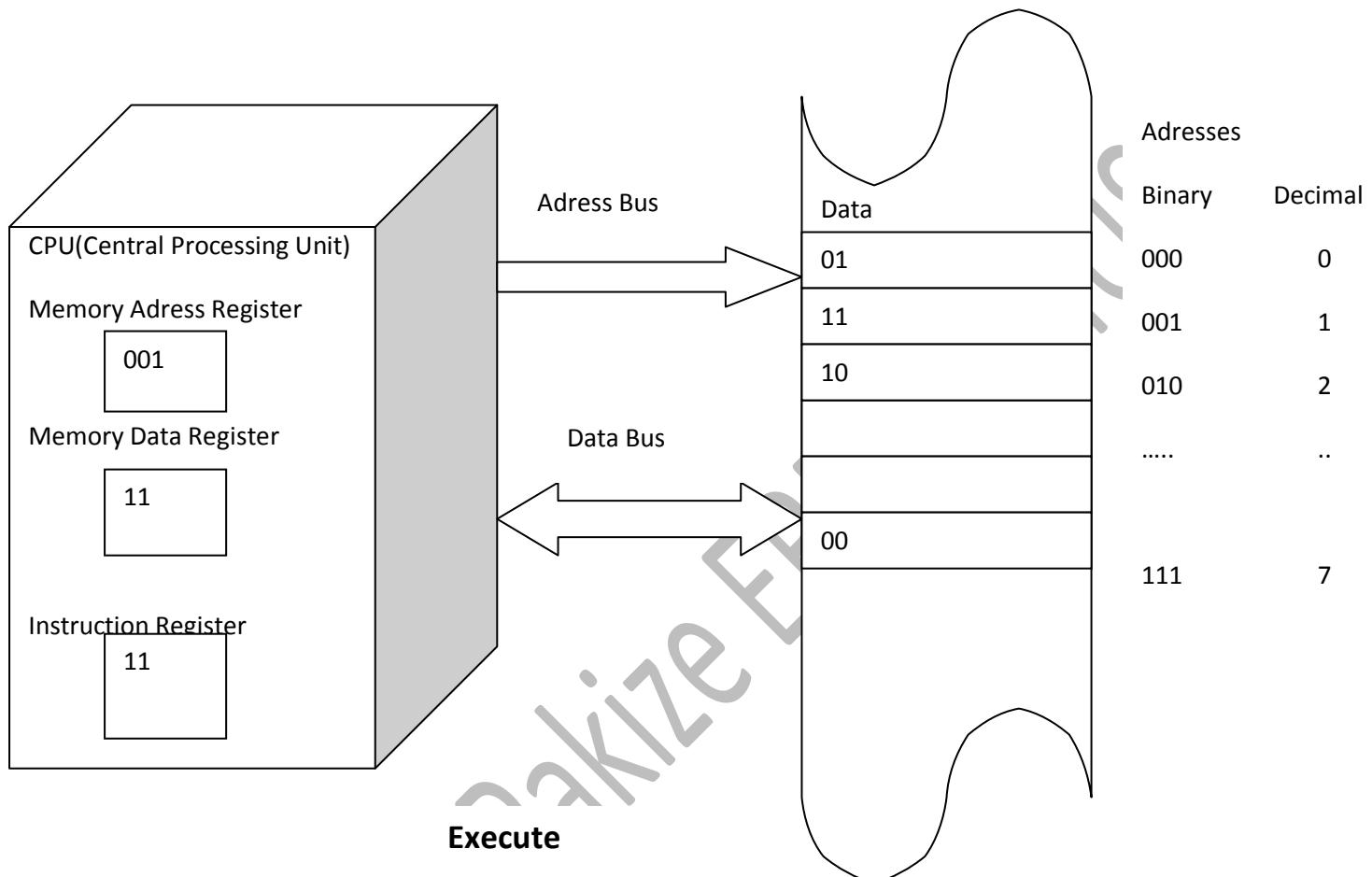
Örnek: Bir sayfasında 25 satır bulunan ve her bir satırda 20 karakter bulunan 30 sayfalık bir kitap bilgisayarda saklanacak olsa kaç byte yer kaplar?

Tablo 1- Veri Birimleri

Veri Birimi Adı	Ölçüsü
Bit	0 veya 1 olabilen bir veri
Nibble	4 bit
Byte	8 bit = 2^3
Word	16 bit = 2^4 = 2 Byte
Double word	32 bit = 2^5 = 4 Byte
Quad Word	64 Bit = 2^6 = 8 Byte
.....
KiloByte(KB)	1024 Byte $\approx 10^3$ Byte = 2^{10} Byte
Megabyte(MB)	1024KB $\approx 10^6$ Byte = 2^{20} Byte
GigaByte(GB)	1024M $\approx 10^9$ Byte = 2^{30} Byte
TeraByte(TB)	1024G $\approx 10^{12}$ Byte = 2^{40} Byte
Peta(PB)	1024P $\approx 10^{15}$ Byte = 2^{50} Byte
Exa(EB)	1024E $\approx 10^{18}$ Byte = 2^{60} Byte
Zetta(ZB)	1024Z $\approx 10^{21}$ Byte = 2^{70} Byte
Yotta(Y)(YB)	1024Z $\approx 10^{24}$ Byte = 2^{80} Byte

Her sayfada 25X20 karakter var ise 30 sayfa de $25 \times 20 \times 30 = 15000$ karakter vardır. Her bir karakter 1 byte ile kodlanırsa $15000 \approx 15$ KByte yer kaplar.





Şekil 1.b Mikroişlemci-Hafıza çalışma mantığı(execute)

Hafıza(Memory) bir kitaplığa benzetelim. Her bir raf farklı bir konu için ayrılsın. Ve aradığımız kitabı ulaşabilmek için kitabı etiketleyelim. 1. Raf, 2. raf,...n. Raf gibi. İşte bir hafızanın adresi, kitapluktaki raf numarası gibidir. Raf sayısı ne kadar çok ise kitaplıkta o kadar çok kitabı olabilir. Bir başka önemli konu ise rafın kapasitesidir. Rafın eni genişliği ne kadar büyük ise o kadar farklı çeşitte kitabı alacaktır. İşte hafızadaki veri uzunluğunu da rafın eni gibi düşünebilirsiniz. Yukarıdaki örnek hafızanın 8 farklı adresi ve 2 bit uzunlığında verileri vardır. Yani 2 bit ile en fazla $2 \times 2 = 4$ farklı veri(kitap) oluşturulabilir. Bu verilerden ise en fazla 8 adet saklayabilirsiniz.

Bir bilgisayarlı sistemde en az bir mikroişlemci ve hafızanın çalışma mantığı yukarıdaki şekillerde anlatılmaya çalışılmıştır. Mikroişlemci çeşitli komutları olan(yani dışarıdan bu

veriyi aldığında bir işlem olduğunu algılayan) ve bu komutların kodunu çözerek, gerektirdiği işlemleri yapan bir entegredir. Bu tanımdan da anlaşılacağı üzere mikroişlemcinin iş yapabilmesi için önce dışarıdan komutlar alması gereklidir. İşte bu komutlarında bir hafızada tutularak sırası ile mikroişlemciye gönderilmesi gereklidir.

Mikroişlemci gerekli enerji ve saat darbesi aldıktan sonra hafiza adres kayıtcısında bulunan adrese gider. Yani program hangi adresten itibaren çalışacak ise o adrese gider. Bu adresteki ilk komutu hafiza veri kayıtçısına getirir.(Fetch) Komutun kodunu çözer. Komutun gerektirdiği işlem gerçekleştirildikten sonra, bir sonraki adresteki veri alınarak işlemler bu şekilde komutlar sona erene kadar (end) devam eder.

Yukarıda da anlatıldığı ve şekil 1.a ve 1.b'de görüldüğü gibi hafizaya yazılı olan her bilgi (data) yaptığı işe göre ingilizce kısaltmalarla kodlara aktarılır. Bu dile de assembly dili adı verilir. Ancak Assembly dili de yetersiz olduğundan daha üst düzey programlar geliştirmiştir.

Üst düzey programlarda geliştirilecek bir programın makine diline dönüştürülmesinde çeşitli süreçler vardır. Önce yüksek düzeyli bir programlama dili kendi editörü ile yazılır. Daha sonra programlama dili ile yazılmış programın yazım hatalarının olup olmadığı kontrol edilerek obje kodu üretilir. Bu aşamaya **derleme (compiling)** adı verilir. Derlenmiş ara kod diğer kütüphane ve parça programlarla birleştirilerek makine dilinde program oluşturulur. Bu işleme de **bağlama(linking)** adı verilir. Ancak bazı programlama dillerinde derleme ve bağlama bir bütündür. Artık makine diline çevrilen .exe uzantılı program icra edilerek, sonuçları görülebilir. Bazı programlama dilleri ise **interpreter(yorumlayıcı)**'ya sahiptir.

Soru: Derleyici ile yorumlayıcı arasında ne fark vardır? Araştırınız

Program çalıştırıldıktan sonra doğruluğu test edilerek her olası sonuç için doğru sonuç verip vermediği kontrol edilir.

Hata Yakalama ve Ayıklama

Bir bilgisayar için analiz aşamasından başlayıp algoritma ve kodlama ile bilgisayara kaydettiğimiz programın hataları var ise bunları tesbit etmek gereklidir. Bilgisayar programcılığında temelde iki türlü hata yapılır.

- 1. Yazım hatası(Syntax Error) :** Yazılan programda programlama dili kurallarına uygun olmayan ifadelerin oluşturduğu hatalardır. Örneğin **for** yerine **fr** yapılması **if** yerine büyük harflerle **İF** yapılması gibi. **Düzeltilmesi kolay hatalardır.** Hatanın yapıldığı satır ve ne olduğu derleyici tarafından rapor edilir. İngilizce bu yardım doğrultusunda hata düzeltılır.

2. **Çalışma zamanı hatası (Run-time Error)** : Programın çalıştırılması sırasında karşılaşılan hatalardır. Programcının göze almadığı veri girişleri veya durumlarda ortaya çıkar.
3. **Mantık hatası(Logical Error)** : Bulunması en zor hatalardır. Derleyicinde uyarı veremediği ama program test edilirken ortaya çıkan hatalardır.
 - a. **Bug** : Mantıksal hatalar bazen **bug** yani **böcek** diye de tanımlanmış olabilir. Bir çok ticari ve amatör yazılımda olabilir. Hatasız bir program yazmak son derece zordur. Bu sebeple profesyonel yazılımlar yeni versiyonları ile hem teknik açıdan daha yeni bir program, hemde mevcut buglardan arındırılmış bir program sunarlar.
 - b. **Debug** : Mantıksal hataları giderebilmek ve yazılımdaki bug'ları bulabilmek için yapılan işlemin adıdır. Program adım adım test edilerek anormal durumlar bulunmaya çalışılır.

2. Algoritmalarla kullanılan İşlemler

Bu bölümde genelde bir algoritma hazırlarken (aslında programlama dillerinde de benzer operatörler kullanılır.) kullanılan aritmetik, mantıksal ve karşılaştırma operatörleri anlatılmaktadır. Ancak daha sonra programlama dillerinde bu işlemlerin operatörleri gösterilecektir. Aynı işi yapan bir operatör farklı programlama dillerinde farklı şekillerde olabilir. Örneğin **eşit değil** karşılaştırma operatörü Matlab'da \sim C++'da $!=$ ve Basic'te \neq şeklinde olmalıdır.

2.1 Matematiksel İşlemler

Aşağıdaki tabloda Algoritma yazarken kullanılan Matematiksel operatörler verilmiştir. Bu operatörler dilden dile değişebilirler.

İşlem	Matematiksel	Algoritma ve Programlama	Örnek	Sonuç
Toplama	+	+	3+9	12
Çıkarma	-	-	5-2	3
Çarpma	X	*	7*3	21
Bölme	÷	/	6/2	3
Üs Alma	a^b	$^\wedge$	2^3	8
Kök alma	\sqrt{a}	$^{(1/2)}$	$25^{(1/2)}$	5
Mod alma	Mod	%	45%4	1

İşlemlerin öncelik sırası aşağıda verilmiştir.

- Parantez
- Üs alma

- Çarpma, Bölme
- Toplama, çıkarma

!!!Eş değer iki işlem var ise bilgisayar soldan sağa doğru sırası ile işlem yapar.

Örneğin

$\frac{(3+5x+x^2)^3 - 7x}{\frac{2}{3} + 5x}$ ifadesi bilgisayarda $((3+5*x+x*x)^3-7*x)/(2/3+5*x)$ şeklinde yazılır.

$\frac{(3+5x+x^2)^3 - 7x}{\frac{2}{3} + 5x}$ (ifadesi ise $(3+5*x+x^2)^3-7*x)/(2/(3+7*x)+5*x)$ şeklinde yazılır.

Not: Bilgisayarda işlem sıralamasını öğrenmek için Matlab'ı kullanınız.

Ödev: $\frac{\sqrt[3]{125}}{\frac{4}{1}}$ işlemini bilgisayar için kodlayınız.
 $\frac{2+3-22 * \frac{2}{4}}{45}$

2.2. Karşılaştırma İşlemleri

Sorgu işlemleri olarak düşünülebilir. Sorgu sonucu evet veya hayır (doğru veya yanlış) olur.

Operatör	Matematik	Bilgisayar
Büyük	>	>
Küçük	<	<
Büyük eşit	\geq	$\geq, =>$
Küçük eşit	\leq	$\leq, =<$

Eşit	=	==, veya =
Eşit değil	\neq	<>veya $\sim=$ veya !=

Öncelik sırası yoktur. Hepsı aynı değerdedir. Karşılaştırma işlemlerinin sonucu sayısal ifade değildir. Doğru(1) veya yanlıştır(0).

Örneğin bir b değişkeninin 2'ye eşit olup olmadığı $b==2$ şeklinde sorulanırken $b=2$ ifadesi bir atama ifadesi olup, b değişkeninin değerini 2 yapar.

Örneğin $a=3, b=5, c=7$ için;

$a>9$ ifadesi bir karşılaştırma ifadesi olup sonucu **y yanlış** tır.

$b==6$ ifadesi sonucu **y yanlıştır**.

$c==7$ ifadesi **doğrudur**.

$(a+b)<10$ ifadesi **doğrudur**.

Soru:D<2 ifadesi için ne söylersiniz?

Matlab'da karşılaştırma ifadesi örneklerini çalışabilirsiniz.

2.3. Mantıksal İşlemler

Operatör	Matematik	Bilgisayar
Not(Değil)	'	Not veya ~
And(Ve)	Λ	And & veya &&
Or(Veya)	V	Or veya veya

Öncelik sırası önce parantez, sonra tümleyen veya değil daha sonra ve veya veya işlemi yapılır.

Tek And($\&$) ve OR(|) iki karşılaştırma ifadesini bağlamaya yarar. Dolayısıyla iki karşılaştırma ifadesinden 1 veya 0 sonucu geleceğine göre 4 farklı durum oluşur. And ve Or işlemlerinin doğruluk tablosu aşağıda verilmiştir.

ifade1	ifade2	Sonuç
0	0	0
0	1	0
1	0	0
1	1	1

AND doğruluk tablosu

ifade1	ifade2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	1

OR doğruluk tablosu

Örneğin $a=5, b=3$ için aşağıdaki ifadeleri inceleyelim.

$(a==0) |(b==3)$ ifadesindeki 1. Karşılaştırma ifadesi yanlışır. (a sıfır değil) Ancak 2. İfade **doğrudur**. $b=3$ 'tür. Dolayısıyla sonuç **doğu**'dur.

Kısaca **VEYA** ifadesindeki karşılaştırma elemanlarından herhangi birisi doğru ise sonuç doğru, **VE** ifadesindeki karşılaştırma elemanlarından her ikisi de doğru ise sonuç doğru olur.

&& ve **||** ifadeleride sırasıyla mantıksal **ve** ve **veya**'dır. Bu durumda ifade1 ve ifade2 bir karşılaştırma ifadesi değil değişkendir. Dolayısıyla iki değişkene bit düzeyinde and ve or işlemi uygulanır.

BÖLÜM SONU SORULAR VE UYGULAMALAR

1. $x=3$, $y=5$, $z=1$ için aşağıdaki işlem sonuçlarının ne olacağını hesaplayınız.

- a) $(x+y^2)*z/x+y+z$ b) $x/(y+z)*2/3*4$
c) $x-2+x*5/2*z+1$ d) $x^{1/2}+5^{2/3}$
e) $x^2(0.5)+8^{(1/3)}$

2. Aşağıdaki karşılaştırma ifadelerinin sonuçlarının ne olacağını hesaplayınız.

$x=3$, $y=5$, $z=1$

- a) $x > 3$ b) $(x+y)/z == 2$ c) $x*y*z < 29$
d) $(x < 3) | (y == 5)$ e) $(x != 4) \& (y > 1)$ f) $x == y + z$

3. Aşağıdaki koşullarda doğru olacak karşılaştırma ifadelerini yazınız.(Değişken ismi say1 ve say2 olarak kullanınız.)

- a) 3'ten büyük sayılar b) 23'ten küçük sayılar
c) 31 ile 48 arasındaki sayılar d) 100'den küçük çift sayılar
e) 45'ten küçük tek sayılar f) 3'e ve 5'e bölünebilen sayılar
g) Her ikiside sıfır olan sayılar h) rakamları toplamı kendine bölünebilen sayılar
i) rakamları toplamı 9 olan sayılar j) 8'e bölünebilen sayılar
k) 7'ye bölünemeyen sayılar

3. Algoritma Hazırlama ve Analizi

Algoritma bir bilgisayar programının hazırlanmasında en önemli adımlardan biridir ve kısaca programın işlem adımlarının sözsel olarak adım adım belirlenmesidir. Eğer bu işlem adımları şekillerle ifade edilirse **akış şeması** olarak adlandırılır.

Algoritma yazmak için geliştirilen özel diller olduğu gibi kendi dilimizde kelime ve yönlendirmeler ile de basit algoritmalar hazırlanabilir. Aslında insan her zaman algoritma ile içli dışlıdır. Örneğin akşamdan ertesi gün için bir plan yaparsınız. Aslında bu planda sizin yarınılarınızın algoritmasıdır.

Örneğin:

1. Saat 7:30'da kalk
2. Eğer hava güzel ise tişort, etek giy değilse kazak, pantolon giy.
3. Okula git.
4. Derse gir.
5. Eğer paran 100TL'den az ise anneni ara.

..... gibi yapacağınız işleri sıralarsınız. Algoritmada Bilgisayar Mühendisliği problemleri veya genel problemleri çözümlemek için bu problemlerin çözüm adımlarını çıkarır. Yukarıdan aşağı çalışır. (Top-to-end) Eğer tekrar edilmesi gereken işlem basamakları var ise program akışı istenilen adımlara da yönlendirilebilir.

Algoritmada ve programlarda kullanılan çeşitli terimler vardır. Sabit, değişken, alt program terimleri en sık kullanılan terimlerdir.

Tanımlayıcı: Sabit, değişken veya alt program tanımlarında kullanılan kelimelere denilir. Tanımlayıcılar isimlendirilirken aşağıdaki kurallara dikkat edilmesi gereklidir.

1. Türkçe karakter kullanılmamalıdır.
2. Boşluk kullanılmamalıdır. Yerine _ kullanılabilir. Örneğin bir değişken ismi **ad soyad** olamaz ancak **ad_soyad** olabilir.
3. Sadece rakamlardan oluşan bir değişken ismi olamaz.
4. Tanımlayıcı ismi bir harf veya _ ile başlar.
5. Tanımlayıcı tanımladığı değişken veya sabitin içeriğini hatırlatan kelimeler olursa programcının işi daha kolay olur. Örneğin toplamları tutan değişken **sum** veya **top**, (Not: top bazen programlama dillerinde reserved keyword olabilir.), çarpımları tutan degisken **carp** veya **mult**, sayac **say** veya **count** olarak adlandırılır.

Örnekler:

Tanımlayıcı ismi	
Şirketi	Geçersiz
Sirket	Gecerli
1a	Gecersiz
a1	Gecerli
Dogum_tar	Gecerli
Ad soyad	Gecersiz

Sabit: Bir bilgisayar programında kullanılan ve değeri program boyunca değişmeyen sayısal veya sayısal olmayan verilerdir. $\pi=3.14$ gibi.

Değişken: Bir programlama dilinde tanımlandığı veri türüne göre program boyu farklı değerler alabilen hafiza alanlarıdır.

Alt program: Program içinde defalarca tekrar eden bir kod kesimi var ise bu kodları defalarca tekrar yazmak yerine bir kez alt program olarak yazılır ve her gerektirgünde çağrırlırlar.

Örnek: Dışardan girilen iki sayının toplamını yapacak bir program algoritması yazınız.

1. Başla
2. Klavyeden say1 ve say2 gir.
3. $toplam=say1+say2;$
4. Ekrana toplam'ı yaz.
5. Dur.

3.1. Algoritmalarda Kullanılan Operatörler

Algoritmalarda ve bilgisayar programlarında kullanılan aritmetik, mantıksal ve karşılaştırma operatörleri geçen derste anlatılmıştı. Bunlara ek olarak algoritmalarda kullanılan ve anlaşılması en zor operatör eşitlik operatöridür.

Matematikteki anlamı ile düşünülmesi durumunda tamamen yanlış sonuçlar üretir.

Eşitlik Operatörü(=) Bilgisayar algoritmalarında ve programlamada aktarma anlamına gelir.

Bir değişkenin içeriğini diğer değişkene aktar anlamında kullanılır.

Asıl işlevi \leftarrow simbolü ile ifade edilebilir. Yani $=\leftarrow$

Bir programda $a=5$ 'in anlamı. $a\leftarrow 5$ 'dir. Yani 5 sayısı a değişkenine aktarılacak demektir.

Örnek: Aşağıdaki algoritmada her satırdan sonra a değişkeni içeriğinin ne olduğunu yazınız.

1. $a=2;$
2. $a=a+5$
3. $b=a-1;$
4. $a=a+b;$
5. $a=a*a$
6. $a=a^{1/3}$

7.dur

İşlem	Önceki değeri	Son değer
$a=2;$	-	2
$a=a+5$	2	$a\leftarrow 2+5=7$
$b=a-1;$	-	$b\leftarrow 7-1=6$
$a=a+b;$	7	$a\leftarrow 7+6=13$
$a=a*a$	13	$a\leftarrow 13*13=169$
$a=a^{1/3}$	169	$a\leftarrow 169/3=58.333$

3.2.Algoritmada kullanılan ifadeler

Sayac: Bir algoritma veya programlama dilinde ilk değerden son değere kadar istenen adım değerinde sıra ile artan bir sayısal ifadeye ihtiyaç var ise bu ifadeye sayaç adı verilir.

$a=a+1$ Birer birer artan bir sayaçtır. Algoritmada her kullanıldığından a'nın önceki değerini 1 arttırır. Ardışık sayılar üretmek için bu şekilde sayaç kullanılır.

$say=say+2$ (2'şer artan sayaç)'a örnek verilebilir. Eğer bu sayacın ilk değeri

$say=0$ verilirse çift sayıları,

$say=1$ verilirse tek sayıları oluşturacağı görülmektedir.

Döngü: Bir algoritma içerisinde n kere veya sonsuz kere tekrar eden program adımlarına döngü adı verilir. Yani algoritmada bazı işlem satırlarının birden fazla sayıda işlendiği duruma döngü adı verilir. Döngü oluşturmak için bir şartla bağlı veya şartsız olarak programın akışını başka satırlara gönderen komutlar kullanılır. Git, şart doğru ise ise git gibi.

Aşağıda verilen 1. Örnekte 3 kez çalışan bir döngü ve 2. Örnekte sonsuz döngü görülmektedir.

ÖRNEK 1:	ÖRNEK 2:
1.a=0 2.a=a+1 3.a>3 ise dur 4.2.Adıma git.	1. a=0 2. a=a+1 3.2.adıma git.

Ardışık Toplama: Bir algoritma içerisinde ardışık veya belli düzene göre artan sayıların toplamı isteniyor ise sayac yardımı ile toplama ataması aşağıda verildiği gibi yapılır.

Toplamı saklayacak değişkene top, sayac değişkenini de say olarak adlandıralım.

top=top+say(Sayac ile arttırılan düzendeki sayılar toplanır)

Örnek: Aşağıdaki algoritmaya göre a,top ve carp değişken içerikleri ne olur?

1. basla
2. a=0;top=0;carp=1;
3. a=a+1;
4. Eğer a>5 ise dur.
5. top=top+a;
6. carp=carp*a;
7. 3. Adıma git

İşlem	Bir önceki değerler	Son değerler
	a top carp	a Top carp

a=0;top=0;carp=1;	-	-	-	0	0	1
a=a+1; a>5 ise dur.	0	0	1	a \leftarrow 0+1 a \leftarrow 1	0	1
top=top+a;	1	0	1	1	top \leftarrow 0+1 top \leftarrow 1	1
carp=carp*a;	1	1	1	1	1	carp \leftarrow 1*1 carp \leftarrow 1
3. Adıma git	1	1	1	1	1	1
a=a+1; a>5 ise dur.	1	1	1	a \leftarrow 1+1 a \leftarrow 2	1	1
top=top+a;	2	1	1	2	top \leftarrow 1+2 top \leftarrow 3	1
carp=carp*a;	2	3	1	2	3	carp \leftarrow 1*2 carp \leftarrow 2
3. Adıma git	2	3	2	2	3	2
a=a+1; a>5 ise dur.	2	3	2	a \leftarrow 2+1 a \leftarrow 3	3	2
top=top+a;	3	3	2	3	top \leftarrow 3+3 top \leftarrow 6	2
carp=carp*a;	3	6	2	3	6	carp \leftarrow 2*3 carp \leftarrow 6
3.adıma git	3	6	6	3	6	6

a=a+1; a>5 ise dur.	3	6	6	a \leftarrow 3+1 a \leftarrow 4	6	6
top=top+a;	4	6	6	4	top \leftarrow 6+4 top \leftarrow 10	6
carp=carp*a;	4	10	6	4	10	carp \leftarrow 6*4 carp \leftarrow 24
3.adıma git	4	10	24	4	10	24
a=a+1; a>5 ise dur.	4	10	24	a \leftarrow 4+1 a \leftarrow 5	10	24
top=top+a;	5	10	24	5	top \leftarrow 10+5 top \leftarrow 15	24
carp=carp*a;	5	15	24	5	15	carp \leftarrow 24*5 carp \leftarrow 120
3.adıma git	5	15	120	5	15	120
a=a+1	5	15	120	a \leftarrow 5+1 a \leftarrow 6 a>5 olduğu için program DURUR.	15	120

En son a=6, top=15 ve carp=120 olur.

Bu algoritmada ne öğrendik?

1. Sayac oluşturmayı. Yani birden istenen sayıya kadar birer birer artan bir değişken atamasını
2. Eğer değişken toplamları depolanacak ise değişkene ilk değer olarak sıfır atanmalıdır.
3. Eğer değişken çarpımları depolayacak ise ilk değer olarak bir atanmalıdır. Aksi halde sonuç hep sıfır çıkar.

Bu algoritma ne yaptı?

Bu algoritma 1'den 5'e kadar sayılar toplamını top değişkeninde 1'den 5'e kadar sayıların çarpımını da carp değişkeninde sakladı.

Örnek:

1. say=1;top=0
2. say=say+2;
3. eğer say>21 ise dur
4. top=top+say
5. say ve top değişkenini yazdır.
6. 2. adıma dön.

Bu program ne iş yapar?

DİKKAT!!!! Toplam yapan değişkene ilk değer olarak sıfır atamayı unutmayın.

Ardışık Çarpım: Bir algoritma içerisinde ardışık ve belli düzene göre artan sayıların çarpımı elde edilmek isteniyor ise sayaç yardımı ile aşağıdaki gibi bir atama yapılır.

carp=carp*say(sayac ile arttırılan düzendeği sayılar çarpılır.)

Örnek:

1. say=0;carp=1
2. say=say+2;
3. eğer say<22 ise 5. Adıma git
4. dur
5. carp=carp*say
6. 2. adıma dön.

Bu program ne iş yapar?

DİKKAT!!: Çarpım sonucunu tutan değişkenlere ilk değer olarak bir vermemeyi unutmayın.

- Yukarıdaki örneklerden de görüldüğü üzere algoritma bilgisayar programları için bir ön hazırlık olup, istenen programın bir iş sırası ile adım adım konuşma dili ile ifadesidir.
- Algoritmada her işlem adımına bir numara verilir. Program akısı istenen adıma yönlendirilerek döngü oluşturulabilir.
- Algoritması yapılmış bir programı yazmak oldukça kolaydır.

SORU: Aşağıdaki programın çalışması sonucu değişken değerleri ne olur?

1. Basla
2. F=1
3. S=20
4. Eğer S<1 ise 9. Adıma git.
5. S=S-3
6. F=F+S
7. F=F+2
8. 4. Adıma git
9. F'yi yaz.
10. Dur

NOT: Bu tür algoritmaları analiz ederken;

1. Kaç değişken kullanılmış? Sayaç olarak kullanılan ve toplayıcı veya çarpıcı olarak

- kullanılan değişkenler hangileridir? Bu değişkenlerin ilk değerleri nedir? tesbit edilir.
2. Sonra algoritmanın durma şartını sağlayan karşılaştırma ifadesi hangisidir? Algoritma karşılaştırma sonucu doğru iken mi duracak, yoksa yanlış iken mi duracak? tesbit edilir.
 3. Algoritmanın durma şartını sağlayan karşılaştırma ifadesinde değişkenin en son değeri(algoritmayı sonlandıracak değer) tesbit edildikten sonra, bir önceki değer ile algoritma test edilerek, işlemlerde hangi değişkenlerin kullanıldığı belirlenir.

Soru: Aşağıdaki algoritmanın çalışması sonucu a ve b değerleri ne olur bulunuz?

1. Basla
2. $a=1$
3. $b=20$
4. Eğer $b < 1$ ise 9. Adıma git.
5. $b=b-2$
6. $a=a+b$
7. $a=a+4$
8. 4. Adıma git
9. a ve b'yi yaz.
10. Dur

Yukarıdaki algoritma da $b=20$ 'den -2 azaltılarak $b<1$ olana kadar ki b değerleri ve b sayısı a da toplanıyor. Bir de a değişkenine döngü sayısı kadar 4 ekleniyor. Bu algoritmada DÖNGÜ sarı ile işaretlenen adımlar arasıdır.

b 18, 16, 14, 12, 10, 8, 6, 4, 2 değerleri için işlem görür. 0 iken $0 < 1$ olduğundan 9. adıma gider ve a, b değerlerini yazar.

Örnek 1. 1'den 1000'e kadar tek sayıların toplamını bulan programın algoritmasını yazınız?

1. Başla
2. $sayi=1; toplam=1;$
3. $toplam=toplam+sayi;$
4. Eğer $sayi>1000$ ise 7. adıma git.
5. $sayi=sayi+2;$
6. 3. Adıma git
7. Ekrana toplamı yaz.
8. Dur.

3.3. Bir Algoritma Hazırlama Aşamaları

Bir algoritma hazırlarken önce bir önceki derste anlattığımız program geliştirmenin ilk aşaması olan, analiz aşamasını tamamamamız gereklidir. Hazırlayacağımız algoritmanın girdileri nelerdir, bu girdilerin ilk değerleri nelerdir, ne gibi veri işleme işlemleri isteniyor, kaç değişkene ihtiyacım var, çıktılarımız neler? Bunlar tesbit edildikten sonra algoritma yazabiliriz. Aşağıda basit bir örnek problem üzerinden algoritma hazırlama aşamaları verilmektedir.

Örnek 1. 5'den 21'e kadar(21 dahil değil) sayıların toplamını bulan programın algoritmasını yazınız?

Analiz: Bu program toplama 5'den 21'e kadar ki sayıları toplayacak.

Dolayısıyla iki değişkene ihtiyaç var.

Birinci değişken sayac olarak kullanılacak. Sayac değişkeninin ilk değeri 5 olacak.

Toplam değişkeni olacak. Bu değişken 5'den 20'ye kadarki sayıları toplayacak.

Toplam değişkeninin ilk değeri 0 (sıfır) olacak.

Birde karar ifadesi olacak. Bir karar ifadesi bir karşılaştırma ifadesi içeren ve programın akışı da bu karar değerine göre ikiye ayrılacak.

Karar ifadesini iki şekilde kurabiliyoruz. Eğer **sayac<durma değeri ise**

Veya

sayac>=durma değeri değil ise

Bu iki ifade de aslında aynı şartta çalışmaktadır.

Örneğin bir a değişkeni olsun.

1. **a>5** ise DUR ifadesi ile

2. **a<=5** değilse DUR ifadesi aslında aynı işi yapmaktadır. Biri diğerinin tümleyenidir.(Demorgan) Bu ikinci ifadeyi **a<6** değilse diye de yazabilirim. Bu üç ifadede denktir.

1. karşılaştırmada durma şartı verilen karşılaştırma sonucunun 1 yani doğru olmasıdır. $a>5$ sonucu doğru olduğunda yani $a=6$ olduğunda algoritma duracaktır.

2. karşılaştırmada ise durma şartı karşılaştırma sonucunun 0 yani yanlış olmasıdır. $a=6$ için $a<=5$ değildir. Dolayısıyla algoritma duracaktır.

Yine 2. İfade $a<6$ değilse şeklinde olsaydı. $6<6$ olmadığı için algoritma duracaktı.

a	$a>5$	$(a<=5)$ 'in değili	$(a<6)$ 'nın değili
1	0	1	1
2	0	1	1
3	0	1	1
4	0	1	1
5	0	1	1
6	1	0	0

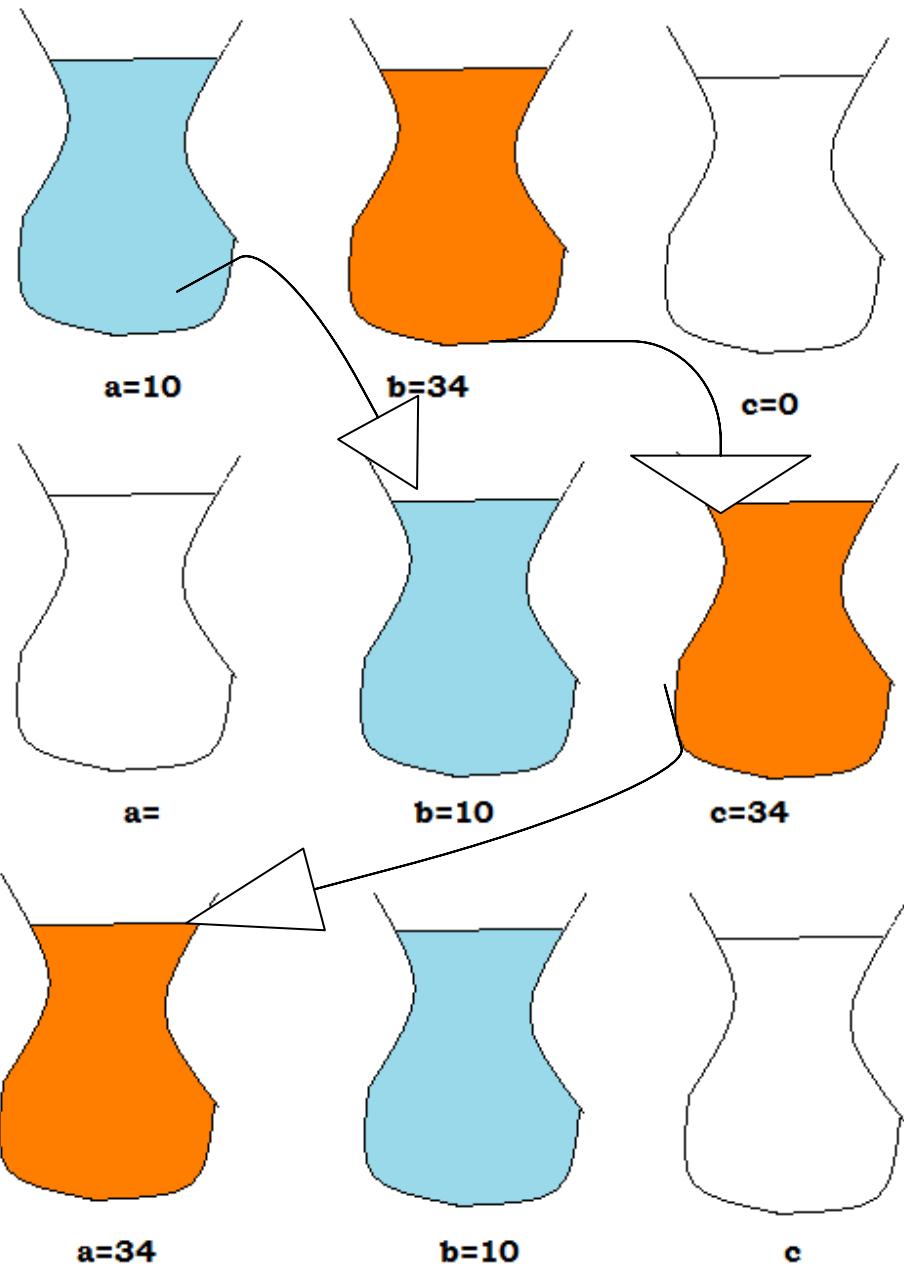
Şimdi yapmak istediğimiz işlemleri sırası ile yazalım. Aşağıda bu işlemlerin sırası değiştiğinde yazılacak algoritmalar verilmiştir.

<ol style="list-style-type: none"> 1. Başla 2. sayac=5 3. toplam=0 4. toplam=toplam+sayac 5. Eğer sayac> 19 ise 8'e git. //sayac>19 karşılaştırması doğru ise 8. Satır'a gider 6. sayac=sayac+1 7. 4. adıma git. 8. Ekrana toplam yaz. 9. Dur 	<ol style="list-style-type: none"> 1. Başla 2. sayac=5 3. toplam=0 4. toplam=toplam+sayac 5. Eğer sayac<20 değilse 8'e git. //sayac<20 karşılaştırması doğru değilse 8. Satır'a gider 6. sayac=sayac+1 7. 4. adıma git. 8. Ekrana toplam yaz. 9. Dur
<ol style="list-style-type: none"> 1. Başla 2. sayac=5 3. toplam=0 4. toplam=toplam+sayac 5. sayac=sayac+1 6. Eğer sayac> 20 ise 8'e git. 7. 4. adıma git. 8. Ekrana toplam yaz. 9. Dur 	<ol style="list-style-type: none"> 1. Başla 2. sayac=5 3. toplam=0 4. toplam=toplam+sayac 5. sayac=sayac+1 6. Eğer sayac< 21 değil ise 8'e git 7. 4. adıma git. 8. Ekrana toplam yaz. 9. Dur

1. Başla	1. Başla
2. sayac=5	2. sayac=5
3. toplam=0	3. toplam=0
4. Eğer sayac > 20 ise 8'e git.	4. toplam=toplam+sayac
5. toplam=toplam+sayac	5. sayac=sayac+1
6. sayac=sayac+1	6. Eğer sayac < 21 ise 4'e git.
7. 4. adıma git.	7. Ekrana toplam yaz.
8. Ekrana toplam yaz.	9. Dur
9. Dur	

Örnek: Dışarıdan girilen a ve b gibi iki değişkenin değerlerini değiştirmek için bir algoritma yazınız.

Bu algoritmayı yazmadan önce aşağıdaki şekli inceleyiniz.



Örnek: Dışarıdan girilen üç sayıdan en küçüğünü bulan bir algoritma yazınız.

Analiz: Kaç değişken gerekir. **Üç değişken sayılar için bir değişken en küçük sayıyı saklamak için olmak üzere dört değişken** gerekir.

Algoritma

1. Başla
2. 1. Sayıyı oku,sayı_1;
3. 2. sayıyı oku, sayı_2;
4. 3. sayıyı oku,sayı_3;
5. en_kucuk=sayı_1;
6. Eğer sayı_2<en_kucuk en_kucuk=sayı_2;
7. Eğer sayı_3<en_kucuk en_kucuk=sayı_3;
8. Yaz en_kucuk
9. Dur

Bu algoritmada 3 değişkene farklı değerler girilerek en_kucuk değerinin ne olduğunu adım adım izleyelim.

Adım	sayı_1	sayı_2	sayı_3	en_kucuk	sayı_1	sayı_2	sayı_3	en_kucuk
1-4	23	34	45		45	12	7	
5				23				45
6		34<23=0				12<45=1		12
7			45<23=0				7<12=1	7
8				23				7

Örnek: Bir sayının hanelerindeki rakamları elde eden program algoritmasını yazınız.

1. Başla
2. 2. sayiyi giriniz?,sayi
3. birler=sayı%10;

4. onlar=int((sayi%100)/10);
5. yuzler=int(sayi/100);
6. Yaz(birler, onlar,yuzler)
7. Dur

Doç. Dr. Pakize ERDOĞMUŞ

BÖLÜM SONU SORULAR VE UYGULAMALAR

1. $\sqrt{\frac{2}{1+x-\frac{3}{y+4-\frac{1}{z}}}}$ ifadesini bilgisayar programlama için kodlayınız. $x=y=z=1$ için sonucu bulunuz.

2. $a=1, b=5, c=9$ için aşağıdaki ifadenin sonucu neolur?

$$a^2/b*c^{(1/2)}=?$$

3. Aşağıdaki algoritmanın çalışması sonucu değişken değerleri ne olur ve b değişkenine bakarak program nasıl bir çıktı üretmektedir açıklayınız?

1. **a=1;b=1;c=0;**

2. **c=b;**

3. **b=b+a+3;**

4. **a=c;**

5. **Eğer b<16 ise 2. Adıma git.**

6. **dur**

4. Üç basamaklı sayılardan rakamları toplamı 10'a eşit olan sayıları toplayan bir algoritma yazınız?

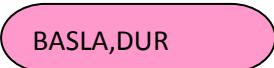
5. 1'den 20'ye kadar olan çift sayıların çarpımını ekrana yazdırın bir programın algoritmasını yazınız?

6. Dışarıdan girilen üç sayı birbirine eşit ise **Hepsi**, iki sayı eşit ise **İkisi** ve hepsi birbirinden farklı ise **Farklı** yazdırın bir programın algoritmasını yazınız.

4. Akış Diyagramları

Bir algoritmanın özel geometrik şekillerle gösterimine akış şeması(flow chart) adı verilir. Akış diyagramlarını izlemek görsel olması sebebi ile algoritmadan daha kolaydır. Akış şemaları her biri farklı bir işlem adımını temsil eden şekillerin oklar ile birleştirilmesinden oluşur. Aşağıda en çok kullanılan akış şeması şekilleri verilmiştir.

- 1. BAŞLA, DUR:** Akış şemasının ilk ve sonuncu şekilleridir.



BASLA,DUR

- 2. VERİ GİRİŞİ:** Klavyeden girilecek veri değerini alır. Birden fazla veri de alınabilir.



Örnek: Klavyeden SAYI1 ve SAYI2 adlı iki değişkenin değerini okur.



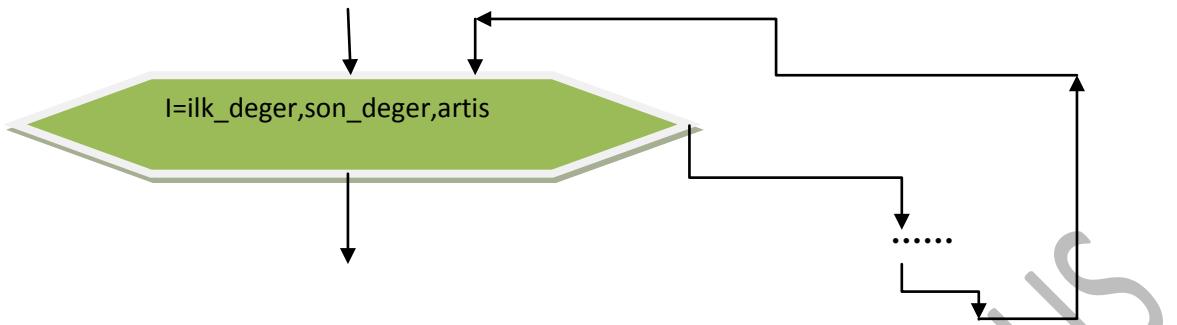
- 3. İŞLEM:** Programın çalışması sırasında yapılan işlemler bu şekilde gösterilir.



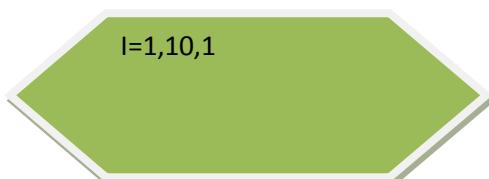
Örnek: Programdaki değişkenin ilk değerini sıfırlar

TOPLAM=0,
CARPIM=1

- 4. DÖNGÜ:** Bir programda ilk değer, artış değeri ve son değer ile belirlenen sayıda işlemin yapılmasını temsil eder. Algoritmada bu işlemin karşılığı birden fazla satırdır.



Aşağıdaki döngü $I=1$ 'den 10'a kadar 1'er artar. 10 kez değer alır. ... ile yukarıda işaretli olan kısım da 10 kez çalışır. Eğer bu kod kesiminde I kullanılıyorsa bu değişen değerleri alır.



$I=-2,10,3$ döngü ifadesinde I değerleri sırası ile $-2,1,4,7,10$ olur. Yani ilk değer ve son değer de işlenir.

$I=-5,2,-3$ döngüsü hiç çalışmaz. Çünkü ilk değer son değerden küçüktür ve azalarak son değere ulaşamaz.

$I=10,5,2$ döngüsünde hiç çalışmaz. 10 İlk değeri 2 artarak 5 son değerine ulaşamaz.

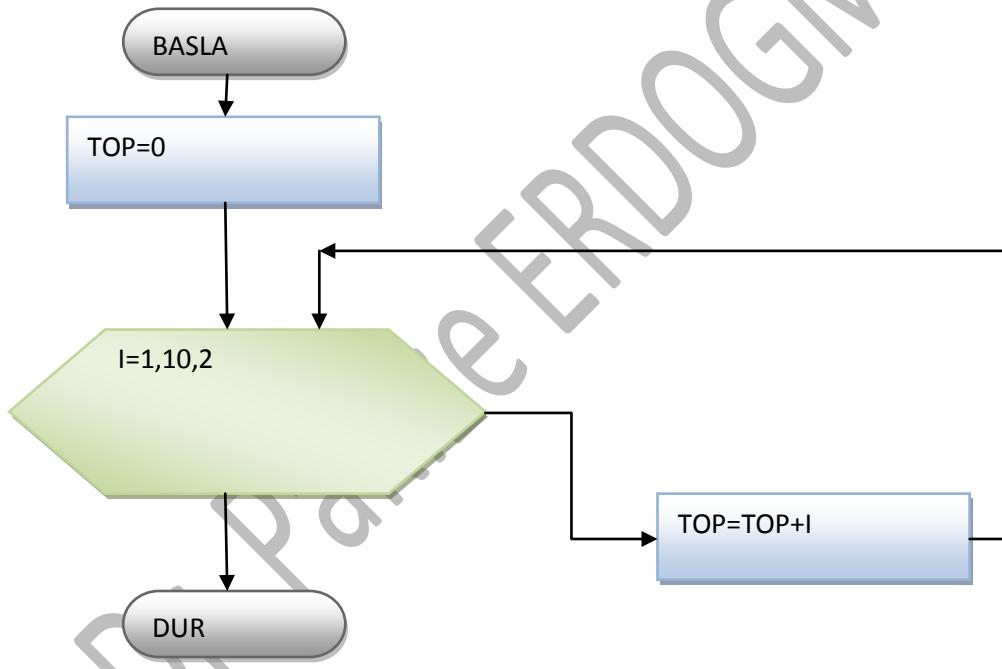
$I=-4,-20,-3$ döngüsünde I değerleri sırası ile $-4,-7,-10,-13,-16,-19$ olur. ... ile gösterilen bu değerler için kisim 6 kez çalışır.

Örnek: Dik kenarları girilen bir dik üçgenin hipotenüsünü ve alanını bulan bir programın akış şemasını çiziniz.

Örnek: Birden 10'a kadar çift sayıları toplayan ve sonucu yazdırın algoritmayı yazınız ve akış şemasını çiziniz.

Algoritma:

1. basla
2. Top=0, I=0
3. Top=Top+I
4. I=I+2
5. I<=10 ise 3. Adıma git.
6. dur



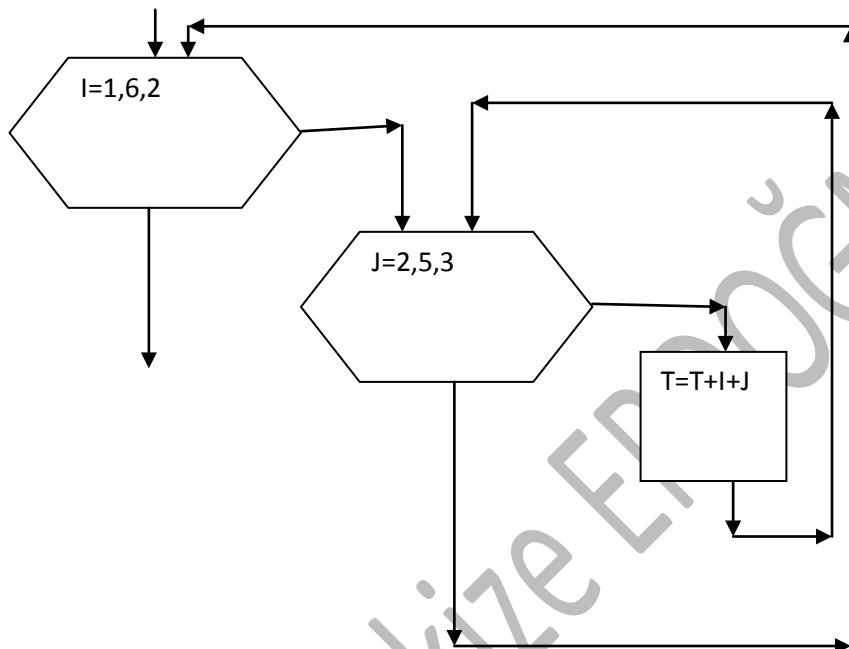
Örnek Uygulamalar:

1. 1'den 10'a kadar sayıların toplamını bulan bir programın akış şemasını çiziniz?
2. 1'den 10'a kadar çift sayıların toplamını bulan bir programın akış şemasını çiziniz.
3. Dışarıdan girilen 10 adet sayıyı toplayan bir program yazınız.
4. Dışardan girilen N değerine göre N adet sayısını ortalamasını bulan bir program yazınız.
5. Dışarıdan girilen x ve y gibi iki sayının toplamını bulan bir program yazınız.

İç içe döngü: Yukarıdak şekli verilen döngü işlevi programlama dillerinde koda karşı gelmektedir. **Acaba bu şekilde iç içe iki tane kullanırsa ne olur?**

Bu durumda dışardaki döngünün her işleyişinde içerisindeki ilk değerden son değere kadar işlem görür ve bu şekilde dışardaki döngünün tekrar sayısı kadar içerisindeki döngü ilk değerden son değere kadar çalışır.

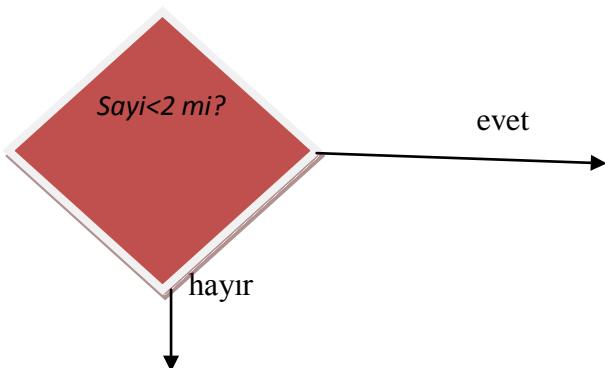
Örnek: Aşağıdaki döngüde T değişkeninin ilk değerinin sıfır olarak verildiğini varsayıarak, her bir adımdaki i,j ve T değerlerini bulunuz.



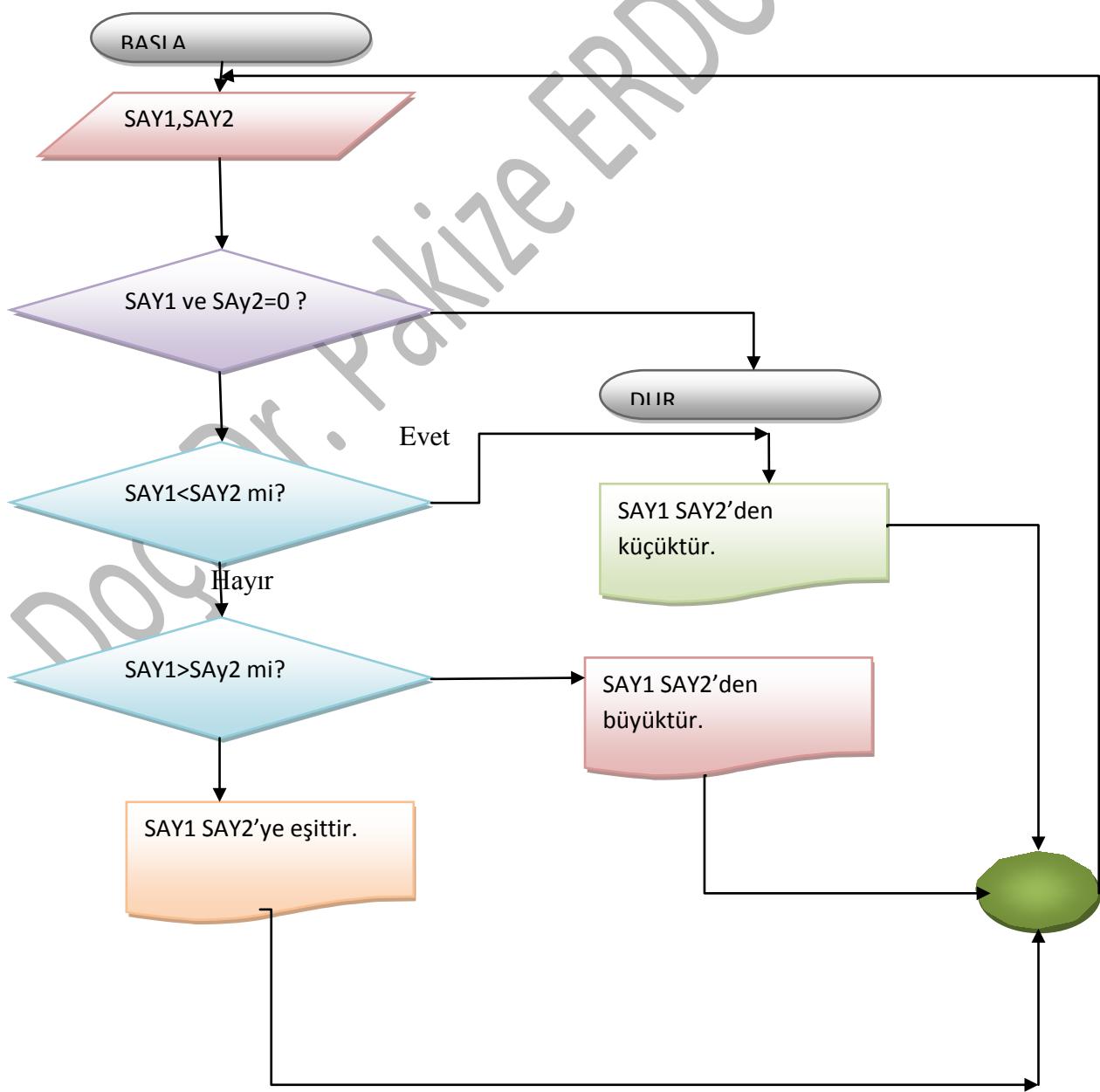
Adım	I	J	T
1.	1	2	=0+1+2=3
2.	1	5	=3+1+5=9
3.	3	2	=9+3+2=14
4.	3	5	=14+3+5=22
5.	5	2	=22+5+2=29
6.	5	5	=29+5+5=39

Aynı şekilde 1'den 10'a kadar çarpım tablosu oluşturabilmek için iç içe 1'er artan iki döngü kullanılır.

5. KARAR: Program akışını karar işlemi sonucuna göre istenen yere dallandırır. Her karar ifadesi program akışını ikiye böler.



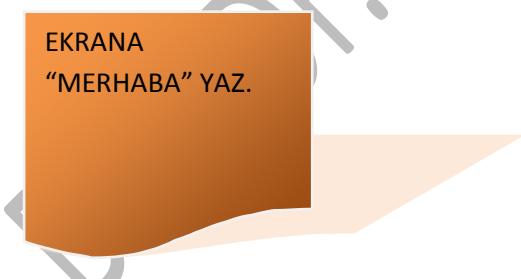
Örnek: Dışarıdan girilen iki sayıyı karşılaştırarak sonuçları yazan ve işlemi her iki sayıda sıfır ise sonlandıran program.



C++ kodu	Matlab Kodu
<pre>#include <iostream> using namespace std; void main() { int say1,say2; say1=say2=1; while ((say1!=0) (say2!=0)) {cout<<"Birinci sayiyi giriniz"<<endl; cin>>say1; cout<<"İkinci sayiyi giriniz"<<endl; cin>>say2; if (say1==say2) cout<<"Esit iki sayı girdiniz"<<endl; else if (say1<say2) cout<<"Birinci sayı daha kucuk"<<endl; else cout<<"İkinci sayı daha kucuk"<<endl; } }</pre>	<pre>clear clc say1=34; say2=45 while say1~=0 say2~=0 say1=input('Birinci sayı'); say2=input('İkinci sayı'); if say1<say2 'say2 büyük' end if say1>say2 'say1 büyük'</pre>

	<pre>end if say1==say2 'say1 say2'ye eşit' end end</pre>
--	--

6. **YAZDIRMA:** Değişken ve işlem sonuçlarını ekrana yazdırma şeklidir.

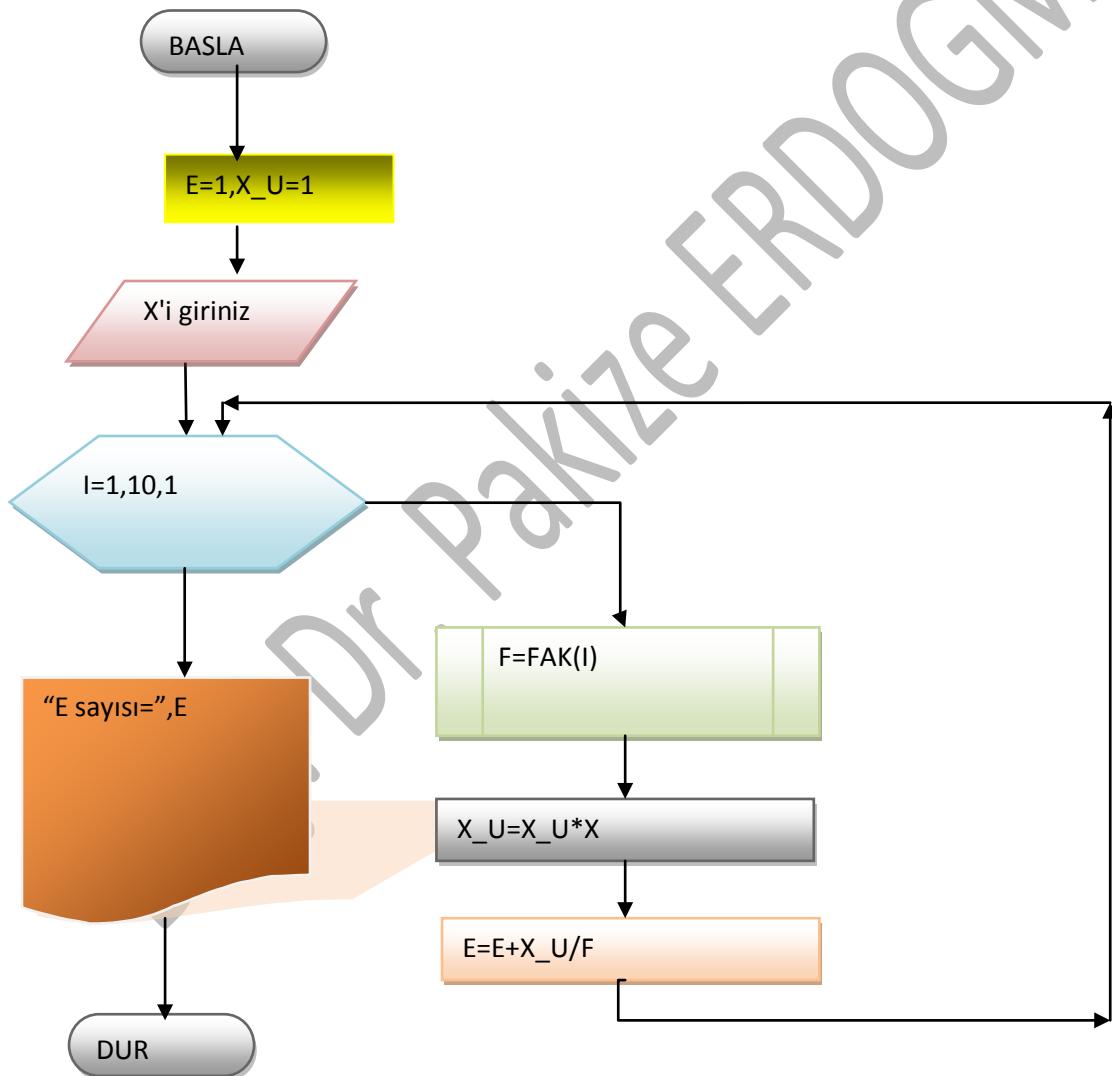


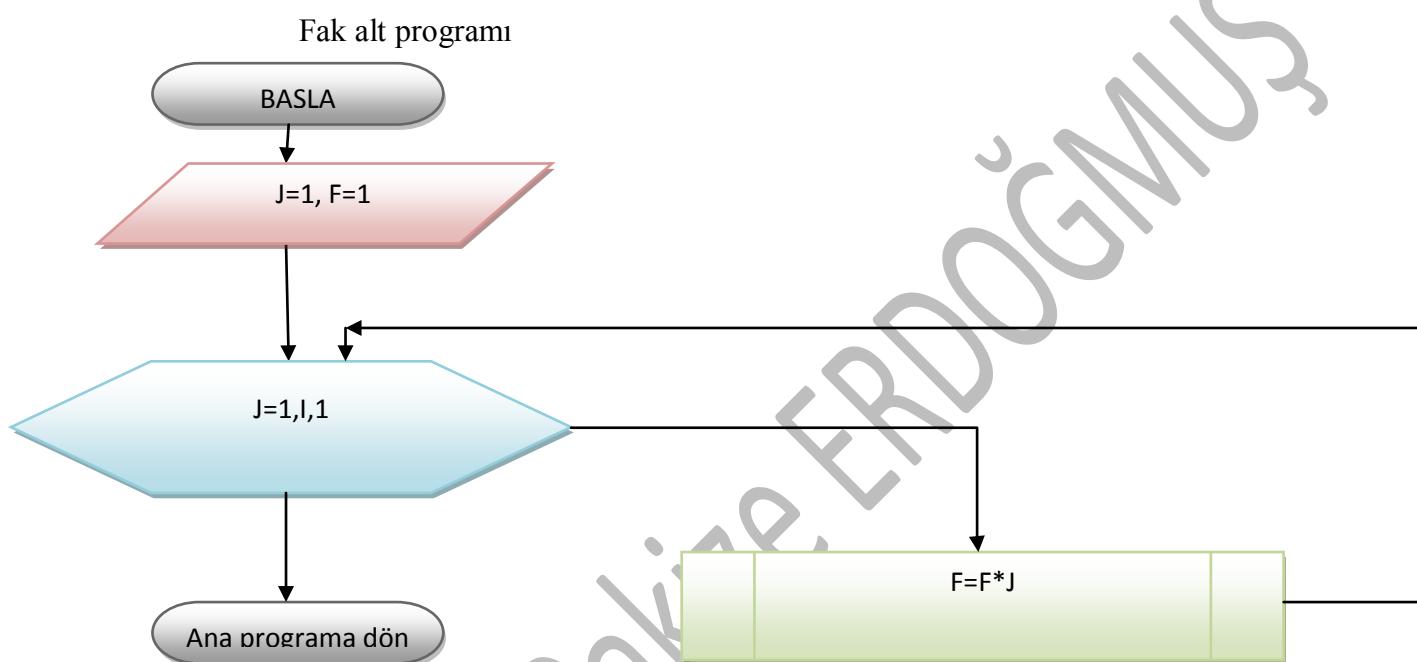
7. **ALT_PROGRAM:** Program içinde tekrar eden işlemlerin bir programcık olarak verildiğini gösteren şekildir.



Örnek: e sayısını 10 terim ile hesaplayarak sonucu ekrana yazan akış şeması.

Ana program:

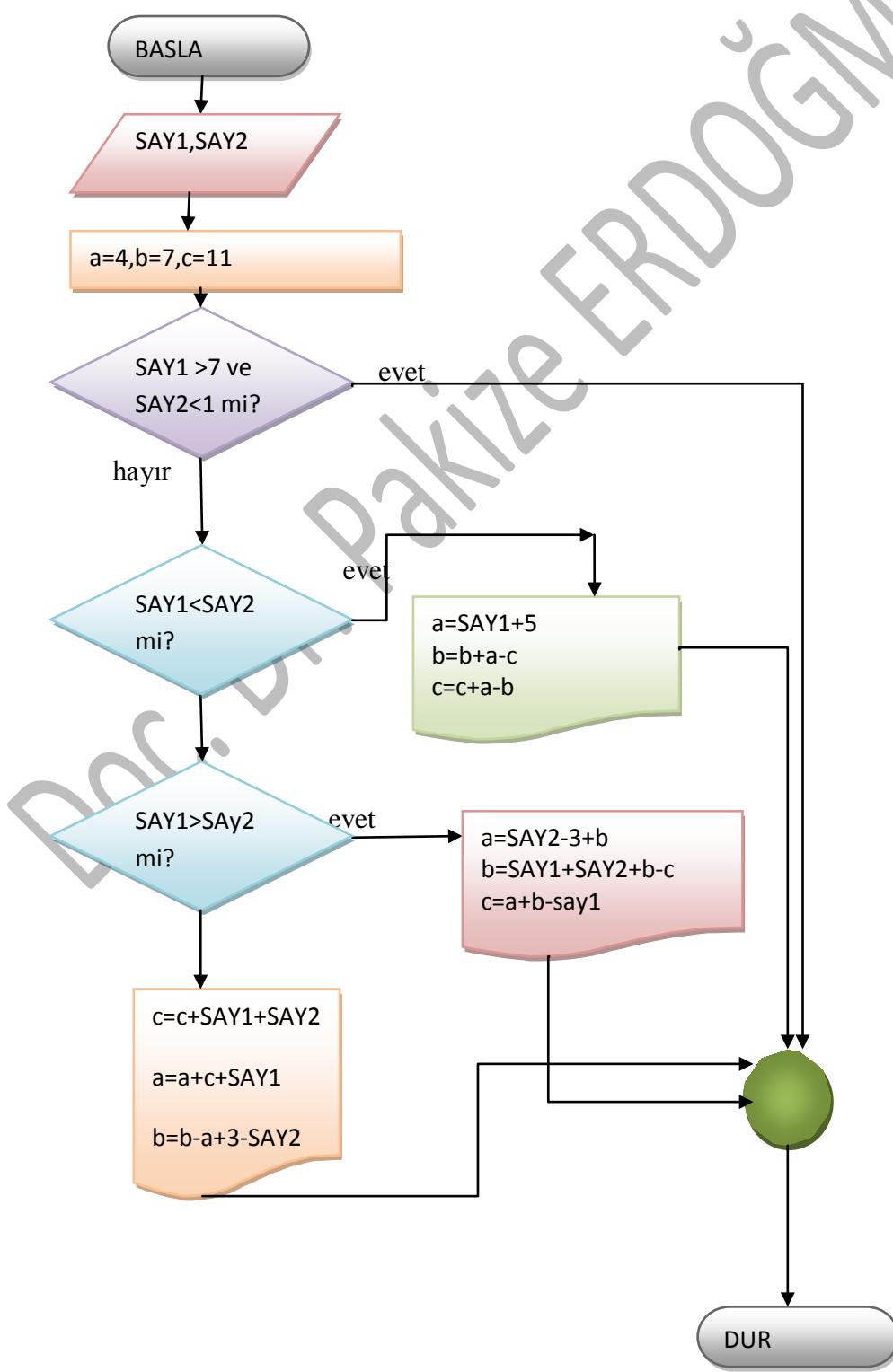




8. BAĞLANTI:

AKİŞ ŞEMASINDA PROGRAM AKIŞININ GİDECEĞİ ŞEKLİ AYNI SAYFADA DEĞİL İSE Veya BİRDEN FAZLA İŞLEM AKIŞI AYNI YERDE TOPLANACAK İSE BAĞLANTI KULLANILIR VE PROGRAM İSTENİLEN ŞEKLÉ YÖNLENDİRİLİR.

Örnek: Aşağıdaki akış şemasına göre dışarıdan SAY1=5, SAY2=3 girilmesi durumunda a,b,c değişken değerleri ne olur?



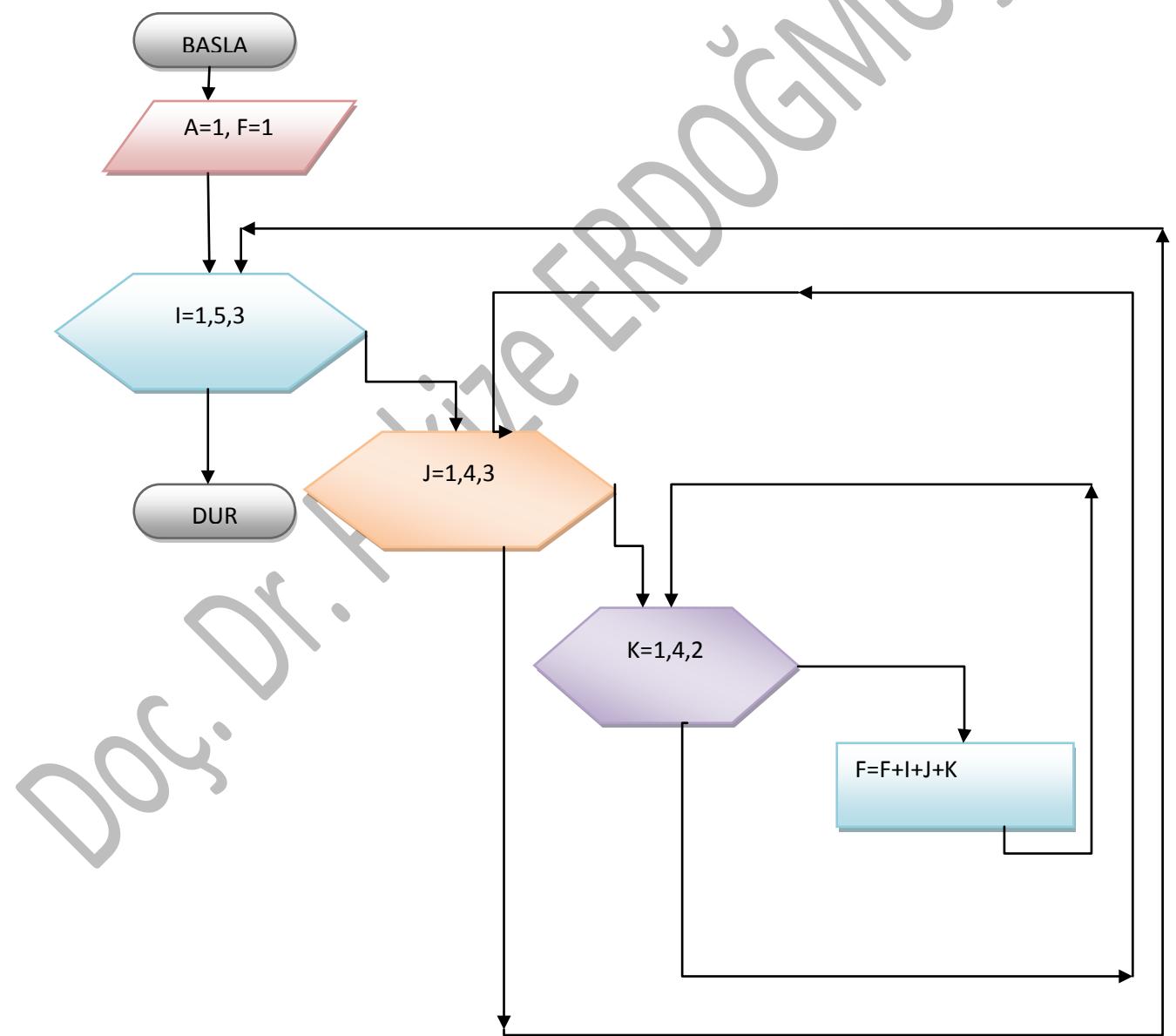
SAY1>SAY2 olduğu için;

$$a=3-3+7=7$$

$$b=5+3+7-11=4$$

$$c=7+4-5=6 \text{ bulunur.}$$

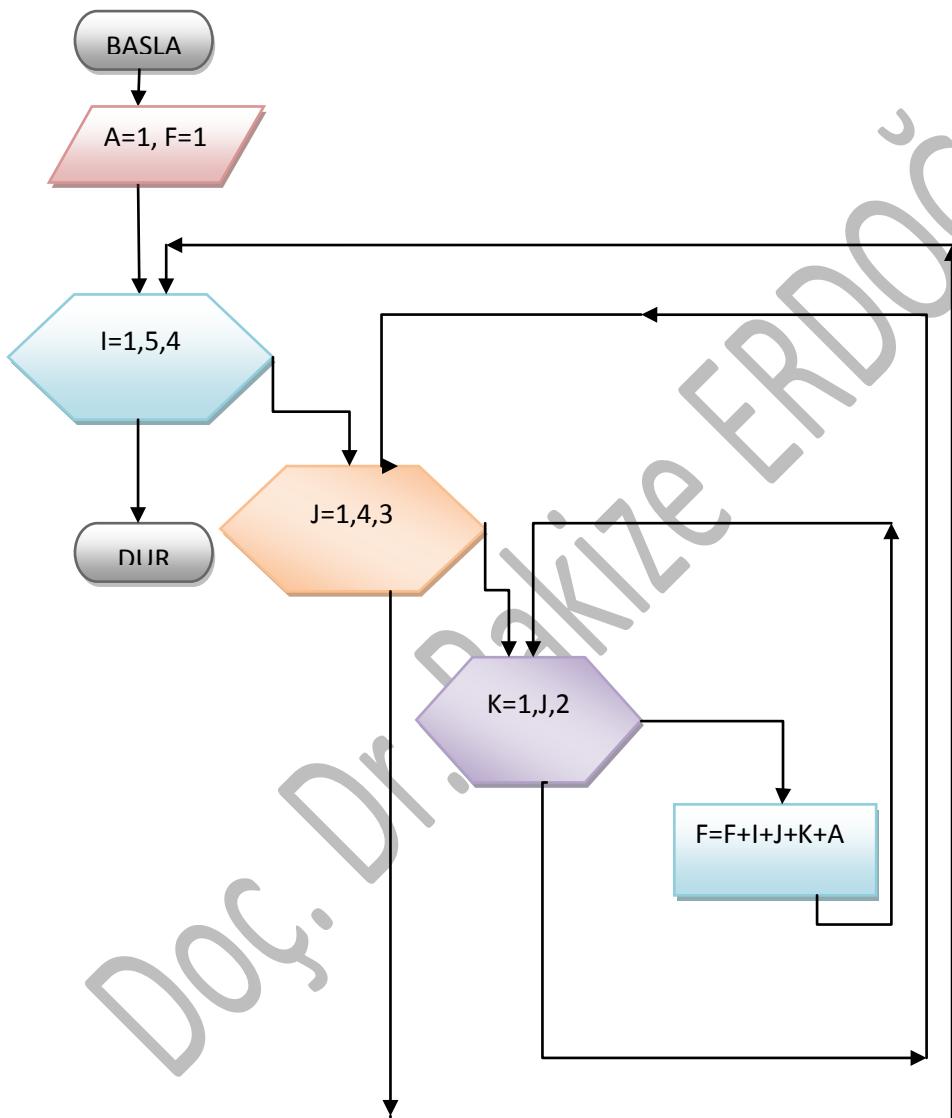
Örnek: Aşağıdaki akış şemasına göre I,J,K, F değerleri ne olur?



Değişken	I	J	K	F
1	1	1	1	1
2				$=1+1+1+1$ $=4$
3			$=1+2=3$	$=4+1+1+3$ $=9$
4		$=1+3=4$	=1	$=9+1+4+1$ $=15$
5			$=1+2=3$	$=15+1+4+3$ $=23$
6	$=1+3=4$	=1	=1	$=23+4+1+1$ $=29$
7			$=1+2=3$	$=29+4+1+3$ $=37$
8		$=1+3=4$	=1	$=37+4+4+1$ $=46$
9			$=1+2=3$	$=46+4+4+3=57$
10				

PROGRAM DURUR				
11	I=4	J=4	K=3	F=57

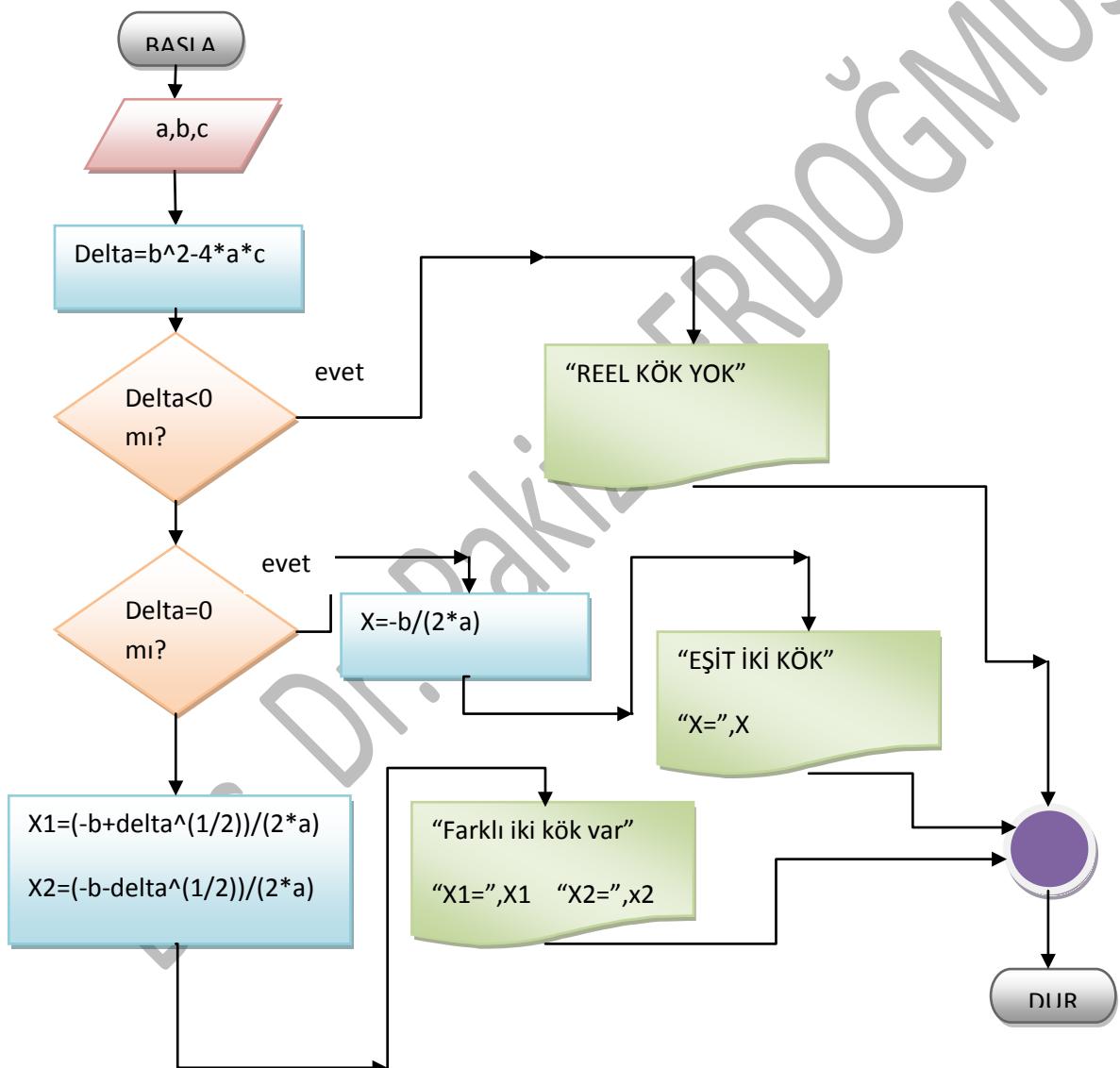
Örnek: Aşağıdaki akış şemasına göre I,J,K, F ve A değerleri ne olur?



Değişken	I	J	K	A	F
1	1	1	1	1	1

2			1		$=1+1+1+1+1 =5$
3		$=1+3=4$	=1		$=5+1+4+1+1 =12$
4			$=1+2 =3$		$=12+1+4+3+1 =21$
5	$=1+4 =5$	1	1	1	$=21+5+1+1+1 =29$
6		$=1+3=4$	=1		$=29+5+4+1+1 =40$
7			$=1+2=3$		$=40+5+4+3+1 =53$
	I=5	J=4	K=3	A=1	F=53

Örnek: 2. dereceden denklem katsayıları a, b, c girildiğinde denklem köklerini bulan programın akış şemasını çiziniz.

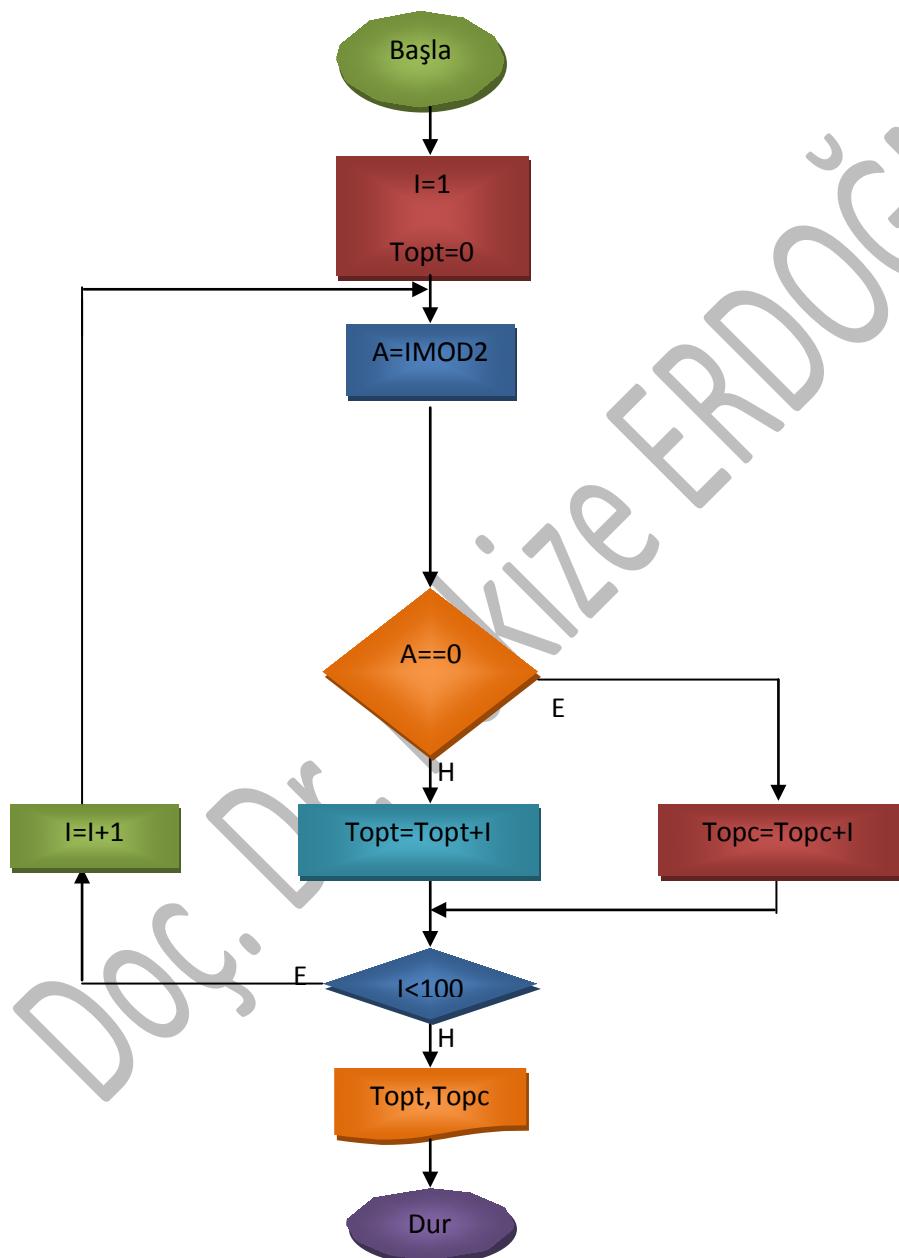


C++ kodu	Matlab kodu
<pre>#include "stdafx.h" #include <iostream> using namespace std; int _tmain(int argc, _TCHAR* argv[]) {float x1,x2,delta; int a,b,c; char sonuc; cout<<"Bu program ax^2+bx+c=0 gibi bir ikinci dereceden denklem koklerini hesaplar"<<endl<<===== =====" <<endl<<"SIRASI ILE a,b,c katsayilarini giriniz"<<endl; cin>>a>>b>>c; delta=b*b-4*a*c; delta>=0?x1=(-b- sqrt(delta))/(2*a):x1=0; delta>=0?x2=(- b+sqrt(delta))/(2*a):x2=0; cout<<"Denklemin 1. koku x1=<<x1<<endl; cout<<"Denklemin 2.koku x2=<<x2<<endl; system("pause"); return 0; }</pre>	<pre>clc clear a=1;b=1;c=1 while a~=0 & b~=0& c~=0 a=input('x^2 katsayisi'); b=input('x katsayisi'); c=input('sabit sayi'); d=b^2-4*a*c if d<0 'Reel kök yok' elseif d>0 x1=(-b+sqrt(d))/(2*a) x1=(-b-sqrt(d))/(2*a) else 'Eşit iki kök' x=-b/(2*a) end end</pre>

Örnek: 1-100 arasındaki tek ve çift sayıların toplamını ayrı ayrı bulan algoritma ve akış diyagramı.

Algoritma:

1. Başla
2. I=1, Topt=0, Topc=0
3. A=IMOD2
4. Eğer A=0 ise Topc=Topc+I ve 6. adıma git
5. Topt=Topt+I
6. Eğer I<100 ise I=I+1 al ve 3. adıma geri dön
7. Topt,Topc değerlerini yaz
8. Dur

Akış Diyagramı:

Matlab kodu

```

%% Tek bir döngü ile tek ve çift sayılar toplamı
tic
top_tek=0;
top_cift=0;
for i=1:100000
    if mod(i,2)==0
        top_cift=top_cift+i;
    else
        top_tek=top_tek+i;
    end
end
sure1=toc
top_tek
top_cift
%% Çift döngü
tic
top_tek=0;
top_cift=0;
for i=1:2:100000
    top_tek=top_tek+i;
end
for i=0:2:100000
    top_cift=top_cift+i;
end
sure2=toc

```

Örnek: Dışarıdan girilen üç sayıyı büyükten küçüğe doğru sıralayan bir algoritma

- 1.Başla
- 2.A,B,C sayılarını giriniz.
3. EB1=A;
4. Eğer B>EB1 ise EB1=B:4. Adıma git
5. Eğer C>EB1 ise EB1=C:6.adıma git
4. Eğer C>A ise EB2=C: EB3=A: değilse EB2=A:EB3=C
- 5.10. Adıma git
6. Eğer B>A EB2=B: EB3=A değilse EB2=A:EB3=B
7. 10.Adıma git
8. Eğer B>C ise EB2=B:EB3=C değilse EB2=C:EB3=B
10. Ekrana EB1,EB2,EB3 yaz.
- 11.Dur.

Örnek: Fibonacci serisinden 100 terim üreterek ekrana yazdırın bir akış şeması çiziniz.

Genel Bilgi: 0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55,89,... sayılarına Fibonacci sayıları adı verilir.

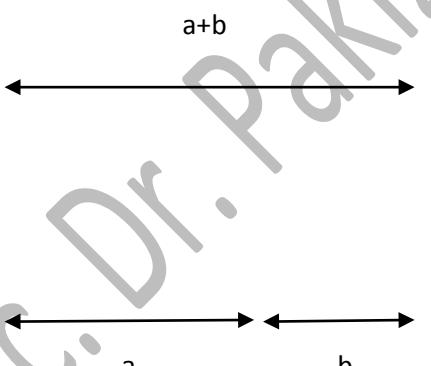
Bu sayılar ilk iki terim dışında son iki terimin toplamı ile elde edilir.

$$F_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F_{n-2} + F_{n-1} & n > 1 \end{cases}$$

Fibonacci sayıları serisinde serinin n. Elemanın n-1. Elemana oranına altın oran(**golden ratio**) adı verilir. $\frac{\sqrt{5} + 1}{2} = 1.6180$

Bilgisayar bilimlerinde dağıtık hesaplama ve paralel işlemede önemli yere sahip olan Fibonacci sayılarının keşfi bilmediğimiz kadar eskiye dayanıyor. Eski Mısır piramitlerinde, Yunan tapınaklarında ve Leonardo da Vinci'nin tablolarında bu oran görülmektedir.

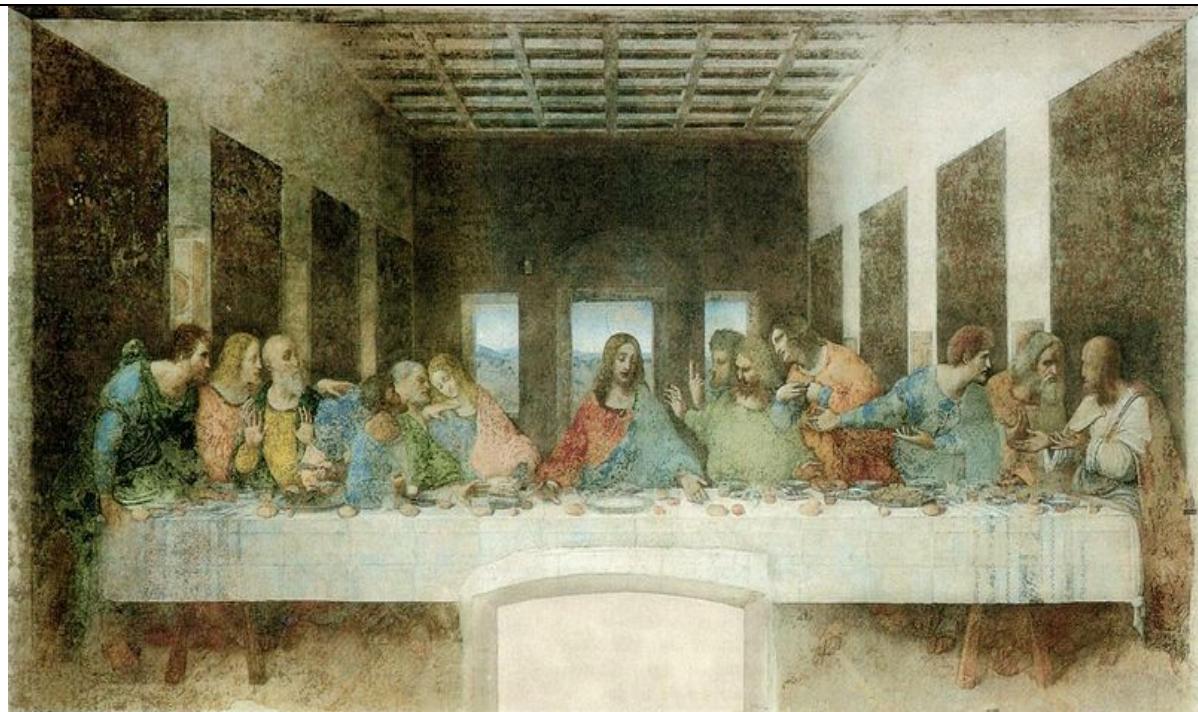
Bu oran kısaca şöyle açıklanabilir. Bir doğru öyle bir noktadan ikiye ayrılsın ki küçük olan doğru parçasının büyük olana oranı, büyük olanın doğrunun tamamına oranı aynı olsun.



$$\frac{a}{b} = \frac{a+b}{a}$$

'dir. $(a/b)=x$ denilerek oran

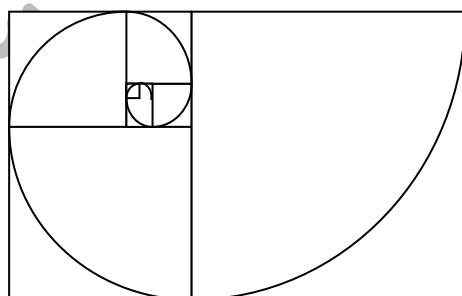
bulunur.



Şekil 2. Leonardo da Vinci'nin altın oranı kullandığı Son Yemek tablosu[2]



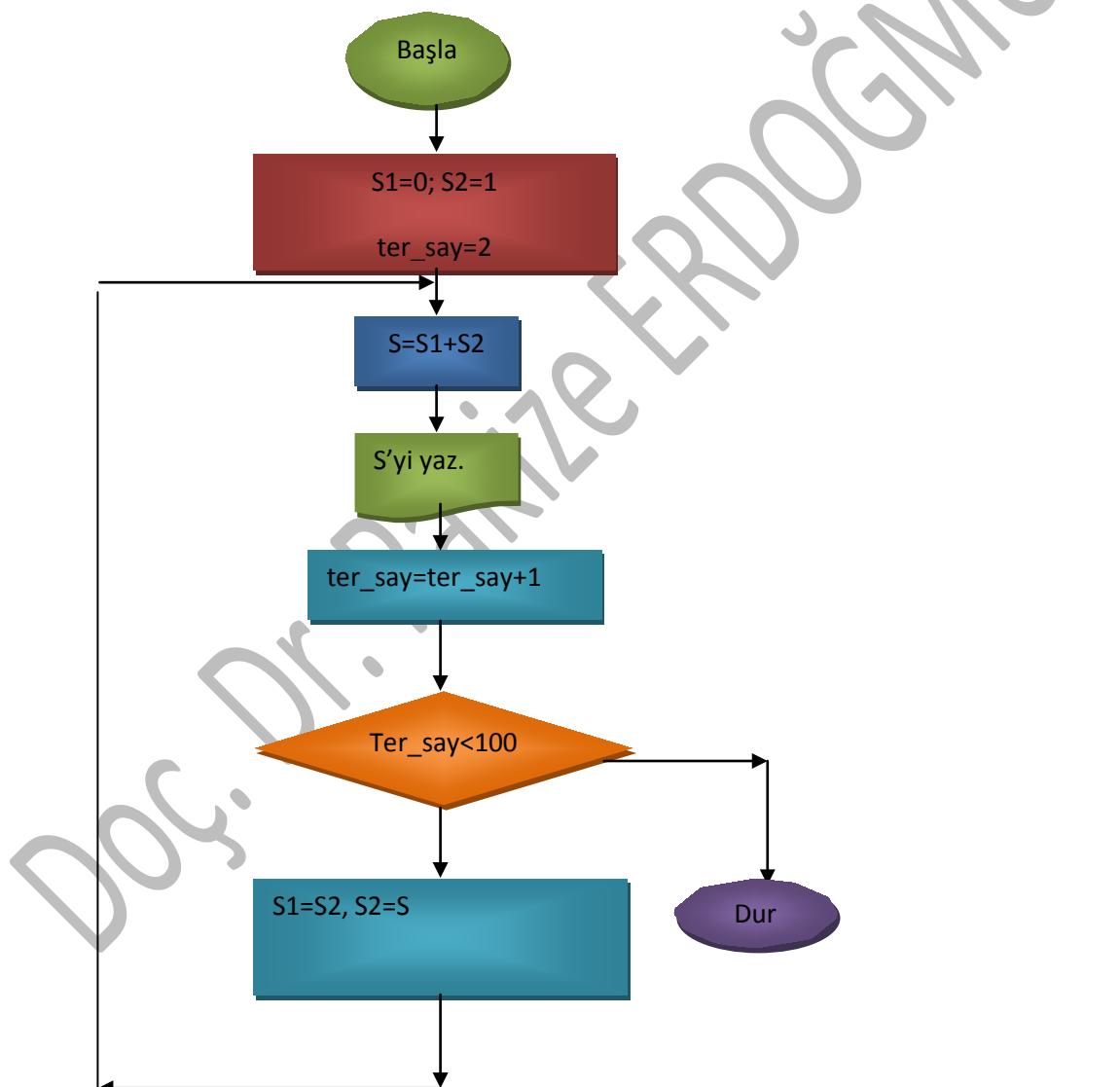
Şekil 3. Lahana yapraklarının içten dışa sayıları da Fibonacci serisine uyar.



Şekil 4. Fibonacci serisinin salyangoz kabuğu görünümü

Algoritma:

1. basla
2. s1=0;
3. s2=1
4. ter_say=2
5. s=s1+s2
6. s 'yi ekrana yaz.
7. s1=s2
8. s2=s
9. ter_say=ter_say+1
10. Eğer ter_say<100 ise 5. Adıma git.
11. Dur.

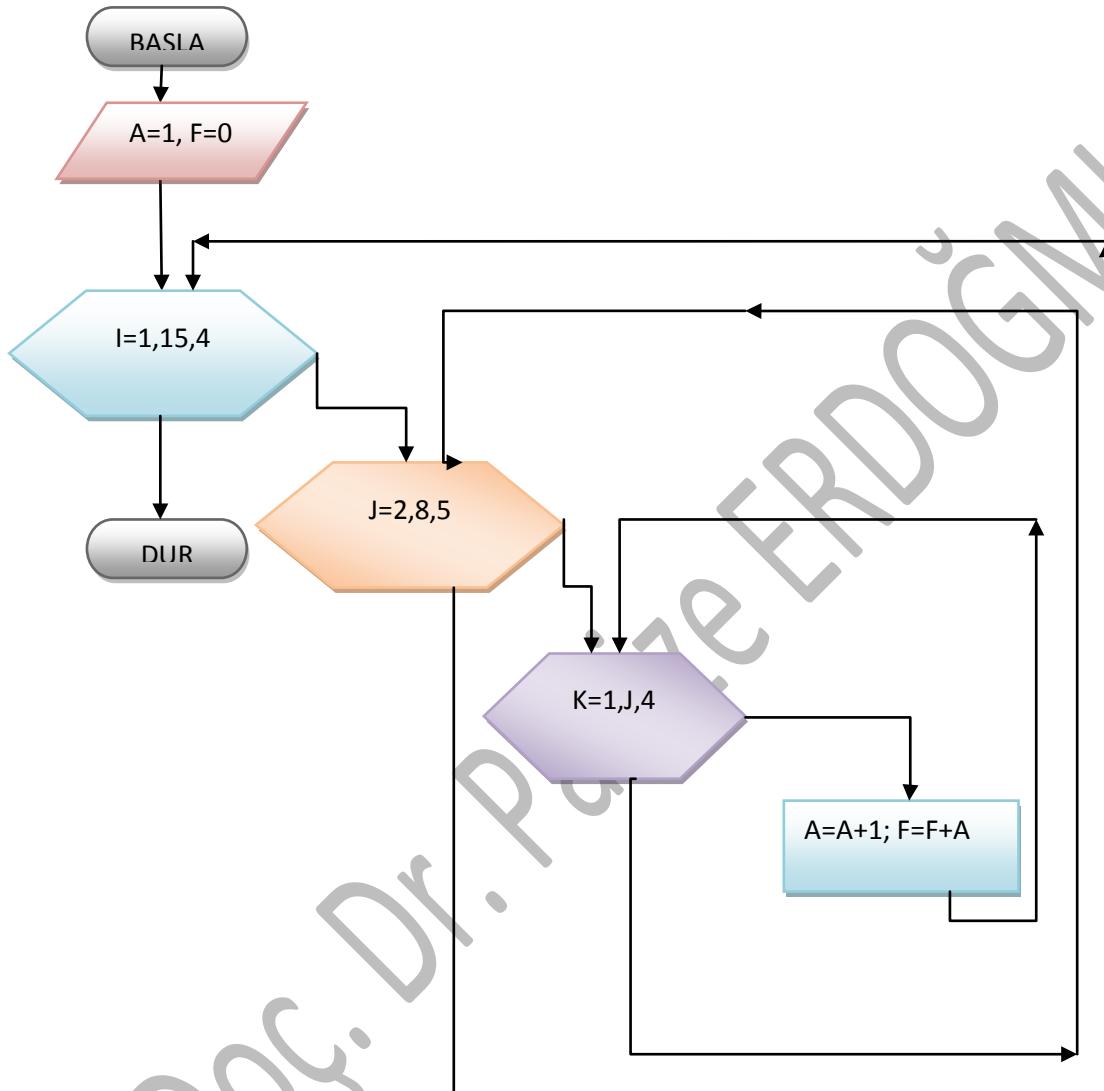


BÖLÜM SONU SORULAR VE UYGULAMALAR

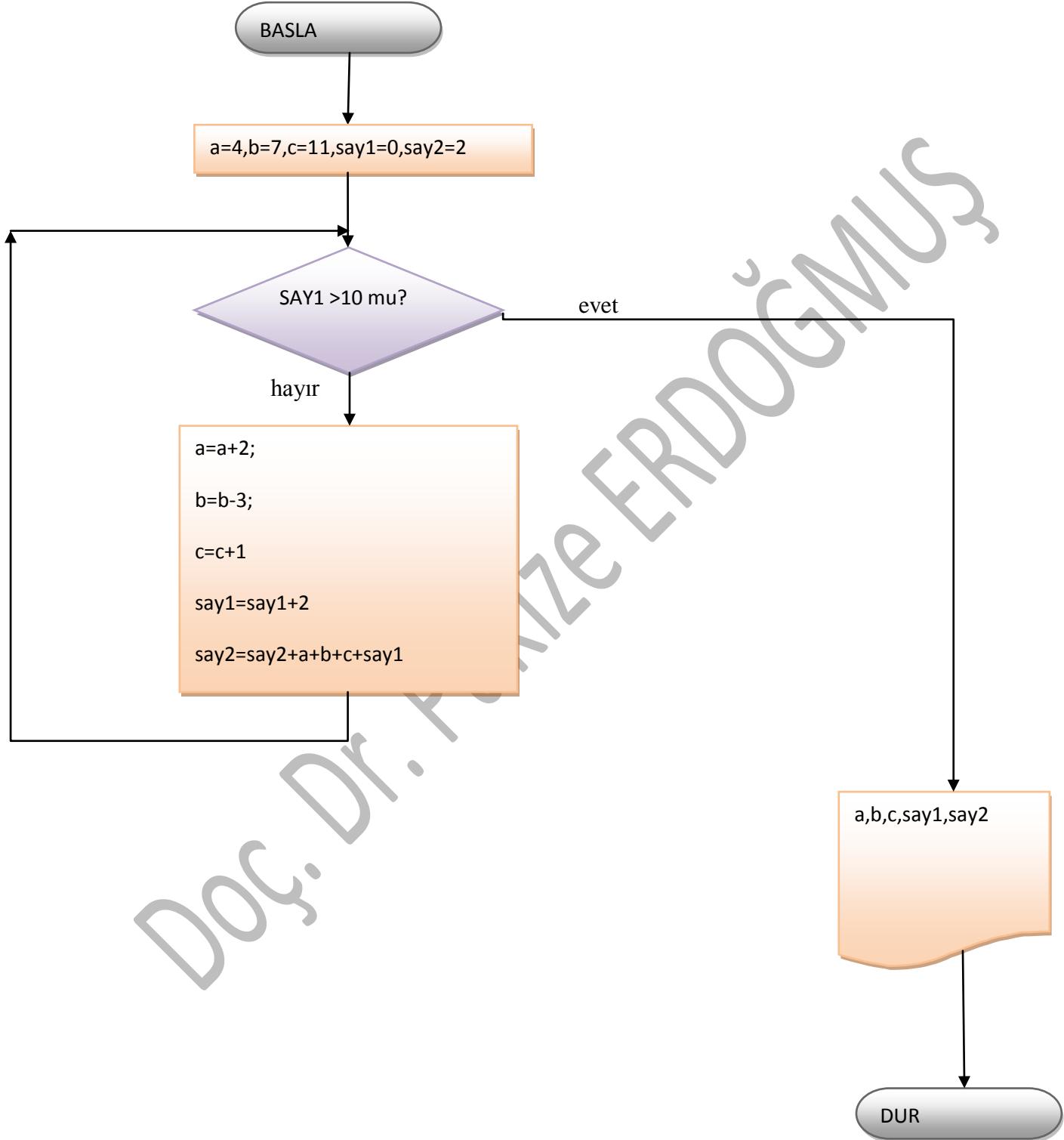
S.1. 100'den 999'a kadar olan sayıların rakam değerlerini toplayan bir akış şeması çiziniz. ($134 \rightarrow 1+3+4=8$)

S.2. 1'den N'e kadar 3 ile tam bölünen sayıların toplamını bulan bir akış şeması çiziniz?

S.3. Aşağıdaki akış şemasına göre A ve F'nin son değerleri ne olur?



S.4. Aşağıdaki akış şeması çalışması sonucu a,b,c,say1 ve say2 değerleri ne olur?



5. Veri Tipleri

Bu bölümde yazacağımız algoritmaları ve oluşturduğumuz akış şemalarını bilgisayar programına dökmek için gerekli alt yapı verilecektir. Bunlardan en önemli altyapı veri tipleri ve operatörlerdir. Bu derste farklı dillerden C++, Basic.NET ve bazen Matlab gösterilecektir. Bilgisayar ile çok büyük miktarlardaki verileri işleyebildiğimizi biliyoruz. Verileri işleyebilmek içinde değişkenlere ihtiyaç duyuyoruz. Algoritma ve akış şemalarında değişkenlere ilk değer vererek kullandık. Oysa programlama dillerinde önce tanımladığımız değişkenin ne tür veri saklayacağını programa bildirmemiz gereklidir. Bu işlem değişken tanımı olarak ta adlandırılır. Aşağıda bazı önemli tanımlar verilmiştir.

Ifade: Her türlü sabit, değişken ve fonksiyonlardan meydana gelen ve program satırında eşitliğin sağ tarafında yer alan kısma ifade denilir.

$a=a+b*c-23$ 'de $a+b*c-23$ bir ifadedir. Bu ifade a değişkenine atanır. Yani önce ifade işlenir sonra değişkene atanır.

Değişken: Kullanıcı tarafından serbestçe tanımlanabilen ve sürekli içerikleri değiştirebilen, sabit olmayan elemanlardır. Bir değişkenin **adi**, **adresi**, **tipi**, **değeri önemlidir** bileşenleridir.

Sabit: Değeri bir kere tanımlandıktan sonra değişimyeni ifadelerdir.

Şimdi veri tiplerine geçmeden önce temel olarak iki tip veri olduğunu hatırlatmamız gereklidir. Önceki derslerimizde de bahsedildiği gibi temel olarak iki türlü veri tipi vardır. Bunlar;

1. Sayısal veriler(3,5,1e89,0.0002,gibi)

Üslü sayılar programlama dillerinde e ile gösterilir.

$2 \times 10^{23} \rightarrow 2E23$ şeklinde gösterilir.

2. Alfayısal(sayısal olmayan) "Ahmet", "05335429728", "02-10-1398" gibi verilerdir.

5.1. VB.NET'te kullanılan veri tipleri

VB. Net'te Dim(Dimension) ile değişken tanımı yapılır. Basic.NET'te kullanılan veri tipleri aşağıda görüldüğü gibidir.

Değişken Tanımlama Kuralları:

- ✓ Değişken ismi anlaşılır olmalı, program okunabilirliğini kolaylaştırmalıdır.
- ✓ Değişken ismi bir harfle başlamalı. Bir rakam veya işaretle başlamaz.

Add1 add2 doğru

Add 2Add yanlış

- ✓ Değişken isimlerinde boşluk bulunmaz.
Adı_soyadı, dogum_tarihi, doğum_yeri => doğru
Adı soyadı, dogum tarihi, doğum yeri => yanlış

- ✓ Değişken isminde sadece harfler, rakamlar ve alt çizgi karakterleri kullanılır. Özel karakterler kullanılamaz.

Alış_tarihi, gör_yer, maas_prim => doğru

Alış-tarihi, #gör_yer, maas&prim => yanlış

- ✓ VB .NET komutları ve hazır fonksiyonları değişken ismi olarak kullanılamaz.

Örnek: Dim, Not, Sub, End gibi

- ✓ Değişken ismi 255 karakter' den fazla olmamalıdır.

- ✓ Veri tabanı v.s uygulamalarında değişken ismi tanımlanırken her anlamlı kelşmenin ilk harfi büyük yazılır. Bu metod Microsoftta çalışan Macar asıllı Charles Simonyi tarafından geliştirilmiştir[2] .

DogumYili

AdıSoyadı

OgrenciKimlikNo gibi.

Değişken Tanımlama

Dim değişkenadı As değişkentipi

şeklindedir.

Örnek: Dim x As Integer

Burada x değişken ismidir. Integer ise değişkenin tipidir. Aynı satırda birden fazla değişken tanımlanabilir.

Örnek: Dim a As Integer, bcd As Long, f As String şeklinde veya

Dim abc, klm, n As Integer şeklinde tanımlanabilir.

Değişkenlere başlangıç değeri verilmezse ne olur?

Veri Tipleri

1. Tamsayı veri tipleri

Adı	Depolama Alanı	Değer Aralığı
Byte	1 Byte	0 ile 255 arasında
Sbyte	1 Byte	-128 ile 128 arasında
Short	2 Byte	-32768 ile 32767
Integer	4 Byte	-2147483648 ile 2147483647
Long	8 Byte	-92230000000000000000 ile 92230000000000000000
Ushort	2 Byte	0-65535
UInteger	4 Byte	0-4 294 967 295
ULong	8Byte	0-18 446 744 073 709 551 615

1. Ondalık sayı veri tipleri

Adı	Depolama Alanı	Değer Aralığı
Single	4 Byte	-3.4028235E+38 ile 1.401298E-45
Double	8 Byte	-4.94065645841246544E-324 ile 1.79769313486231570E+308
Decimal	12 Byte	7.9228..... noktadan sonra 29 belirli haneyi tutabilir

2. Boolean

True veya False değeri alabilir. Değişkenlere değer True yada False olarak atanabilir. Veya atanın sayı 0 ise False değilse True kabul edilir.

Bu tipte değişkenler işleme girerken True ise -1 olarak, False ise 0 olarak işlem görürler.

Örnek: Dim i As Boolean, j as integer

3. String veri tipi

Karakter sınırı verilmemezse 2 milyara kadar karakteri tutabilirler. Bu tipte değişkene değeri aktarılırken “” çift tırnak arasında atanır.

Örnek: Dim Adı As String="Ali" veya

Adı="Ali"

4. Char Veri Tipi

2 Byte yer kaplar. Tek karakter saklar.

Dizi Değişkenler

Visual Basic.NET'te dizi indis 0'dan başlar. Bir dizi tanımlamak için Dim(Dimension=Boyun) deyimi kullanılır.

Dizi tanımlama özel işareti ()'dir. Eğer dizi değerleri değişken tanımı satırında verilecek ise {} sembollerinin arasında virgül ile ayrılarak verilir.

Örneğin değerleri 1,3,5,9 olan bir sayı dizisi tanımlamak için

Dim sayilar() as integer ={1,3,5,9} şeklinde bir tanım yapılır.

5 elemanlı bir sayı dizisi için Dim sayilar(4) tanımı yeterlidir. Bu tanımdan sonra hafızada; sayilar(0), sayilar(1),sayilar(2),..sayilar(4) olarak 5 yer ayrılmış olur.

$$A = \begin{bmatrix} 2 & 2 & 5 \\ 7 & 8 & 9 \\ 1 & 2 & 3 \\ 4 & 4 & 6 \end{bmatrix}$$

matrisi ise iki boyutlu bir dizi ile tanımlanır.

Bu dizi satır sayısı 4, sütun sayısı 3 olan bir dizi şeklinde tanımlanır. Dim A(3,2) şeklinde yer ayrıılır veya ilk değerler ile birlikte;

Dim A(,)={ {2,2,5}, {7,8,9}, {1,2,3}, {4,4,6} } şeklinde tanımlanır.

Diziye elemanlar {} küme parantezleri içinde verilir ve kullanılırken indis numaraları belirtilerek kullanılır. DiziAdı(3) dizinin 4. elemanını verir.

Örnek: Dim aylar() As String={“Ocak”, ”Şubat”, ”Mart”, ”Nisan”}

aylar(2) ‘sonuç:Mart

Eğer VB.Net’te tanımlı değişkenler ile işlem yaparken değişkenin üst sınırı aşılırsa Örneğin Byte türünde tanımlı bir değişkene 280 değeri atanırsa;

OverFlow Exception was Handled hata mesajı verilir. Yani değişkende taşıma olduğunu bildirir.

6. Programlama Dilleri ve C++ ile Programlama

Programlama dilleri, makine dilinin üzerinde yer alır. Komutları programcının anlayacağı seviyede ingilizce deyimlerden oluşur. Bu dersin kapsamında C++ programlam dili anlatılacaktır.

C++ programlama dili C'nin özelliklerini kullanan fakat C'den farklı bir dildir. C komutları C++'da da çalışır. Fakat C++ komutları C'de çalışmaz.

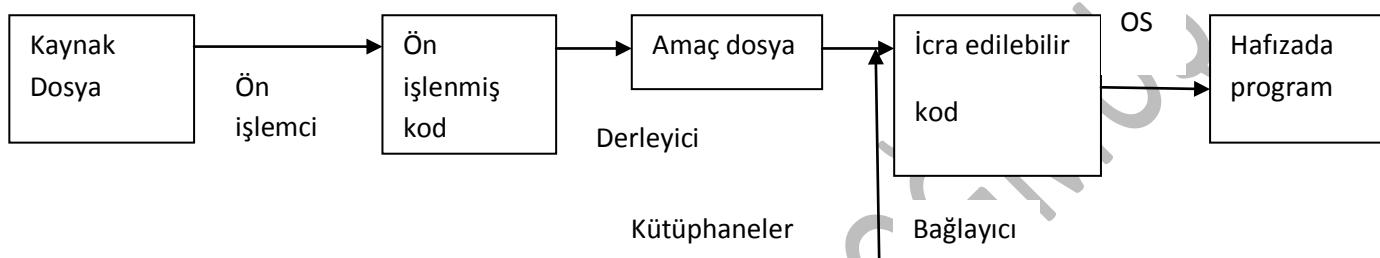
- C bugün kullandığımız birçok yazılım teknolojisinin temelini oluşturur. 1972'li yılların başında Bell Lab'da Dennis Ritchie tarafından geliştirilmiştir.
- C dili 1970'te Key THOMPSON tarafından geliştirilen B dili üzerine kurulmuştur.
- C yapısal bir programlama dilidir.
- ANSI C ve Turbo C olmak üzere iki yazılıma sahiptir. ANSI C, 1970 yılında geliştirilen C'nin ANSI(American National Standard Institute) tarafından standartlaştırılmış halidir. Turbo C ise Borland firması tarafından geliştirilmiştir.
- C birçok programlama dilinin alt yapısını oluşturur.
- C, C++,C# gibi versiyonları vardır.

C++ C'nin genişletilmiş bir biçimidir. C++ ise Bjarne Stroustrup tarafından geliştirilmiştir. Tasarım 70'li yılların ikinci yarısından başlanmış olsa da bütün dünyada yaygınlaşması ve kabul görmesi 80'li yılların sonlarına doğru mümkün olmuştur. Nesne yönelimli programlama(Object Oriented Programming) tekniği(NYP) özellikle büyük kodların üstesinden gelebilmek amacıyla tasarlanmıştır.

C++ derleyicileri C derleyicisini de içermek zorundadır. Yani C++ derleyicisi demek hem C hem de C++ derleyicisi demektir. Derleyici dosyanın uzantısına bakarak kodun C'de mi yoksa C++'ta mı yazılmış olduğuna karar verir. C'de ise uzantısı.c, C++'ta yazılmışsa uzantısı .cpp'dir.

C++ en güçlü programlama dillerinden olup, bir C++ programının çalışması için birçok kütüphaneye ihtiyaç vardır. Bunların bir kısmı otomatik olarak programlara eklenir. Ancak bu kütüphanelerin yetersiz olduğu durumlarda kendi kütüphanelerimizi kendimiz yazabiliriz.

C++’da bir programın Executable hale gelmesi aşamaları aşağıda verilmiştir.



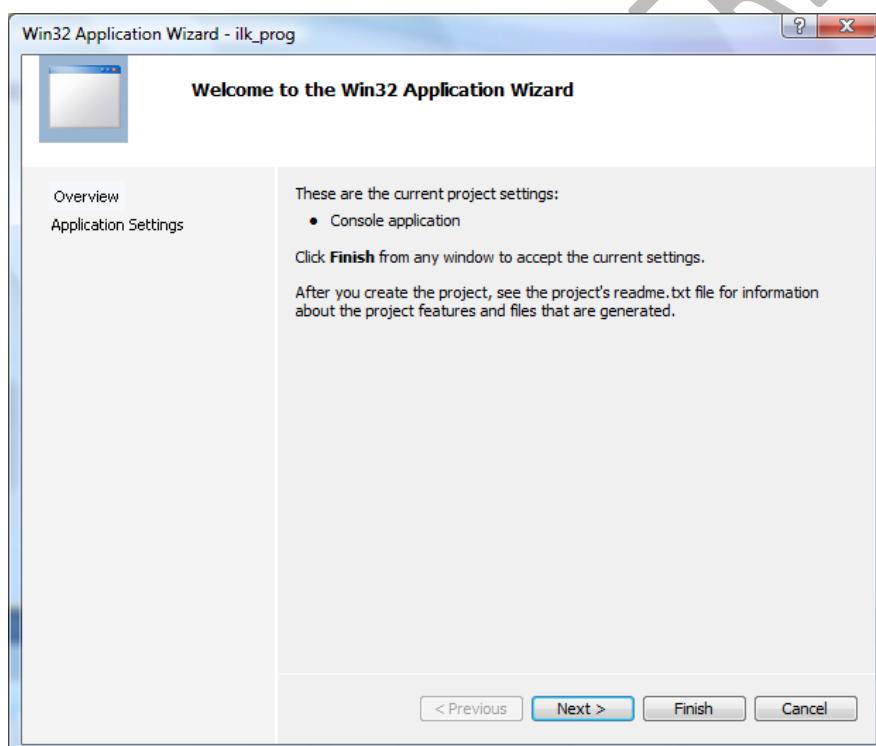
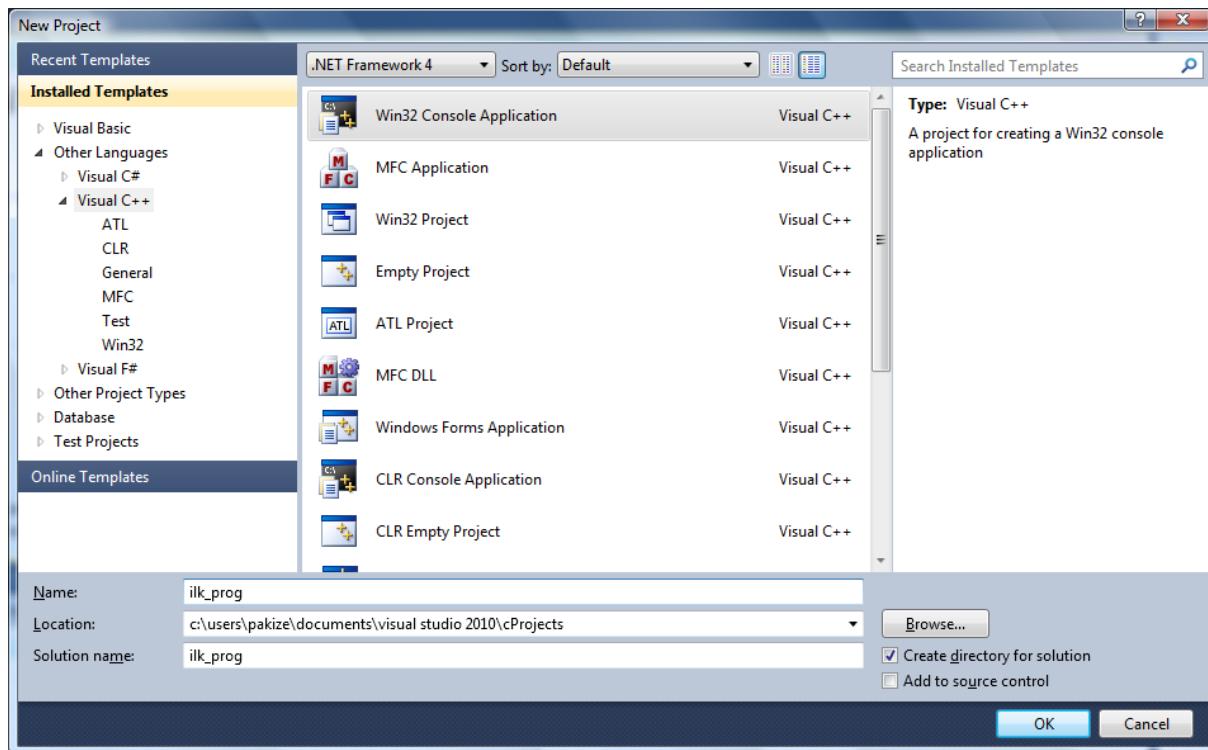
Şekil 5. Bir C++ programının derlenmesi ve çalıştırılması aşamaları

6.1. Microsoft Net C++ ile çalışmak

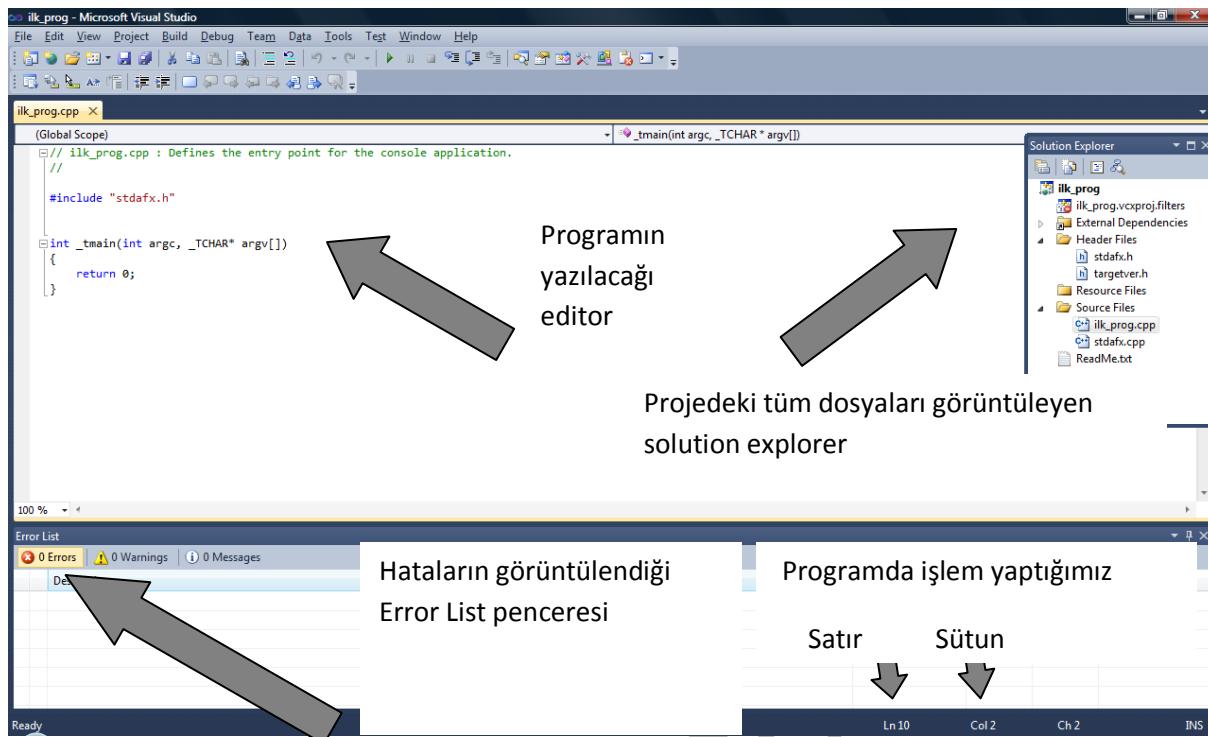
Microsoft .NET’i bilgiyi, bilgi sistemlerini ve cihazları birbirine bağlamak için geliştirilen yazılım teknolojileri kümesidir. NET’in temel anlayışı platformdan bağımsız çalışabilecek programlar üretmektir.

Visual Studio ile bir C++ programı yazmak için programlardan Visual Studio 2010 seçildiğinde aşağıdaki ekran gelir.

File→New Project seçildiğinde gelen aşağıdaki ekranın **Win32 Console Application**’ı seçerek ve dosya adı vererek(ilk_prog) verdığımızda ekrana Win32 Application Wizard-ilk_prog başlıklı bir pencere gelecektir. Burda **Finish** butonu onaylanırsa artık ilk program için gerekli editor açılmış olur.



Programın yazılacağı Studio C++ editor ortamı (IDE) aşağıda görüldüğü gibidir.



Bu IDE ortamında error list açık değil ise View menüsünden açılabilir. Aşağıdaki program yazıldıktan sonra Build menüsünden → Build ilk_prog seçilirse program derlenerek çalıştırılır ve ekran çıktısı aşağıdaki gibi görüntülenir.

The terminal window shows the command: `C:\Users\pakize\Desktop\ilk_prog\Debug\ilk_prog.exe`. The output is:

```
merhaba
Devam etmek için bir tuşa basın . . . -
```

Yukarıdaki program incelenecək olursa ilk iki satır açıklama satırıdır.

C++'da açıklamalar // ile yapılır. Aşağıda açıklama satırına örnekler verilmiştir.

// C++ dosya uzantısı .cpp'dir.

// include ile programın kullandığı kütüphaneler eklenir.

C++ hafızanın etkin kullanılması ve çok hızlı çalışan uygulamalar için sahip olduğu tüm kütüphaneleri kullanmaz. Kullanıcı o program için hangi kütüphanelere ihtiyaç duyarsa, o kütüphaneyi **include** komutu ile programa dahil eder.

```
#include "stdafx.h" //Visual Studio'nun kullandığı kütüphane.  
#include <iostream>  
// iostream giriş çıkış komutlarını içeren bir kütüphanedir.  
  
//ana program gövdesi int main ile başlar  
  
// Ana program İki küme işaretri arasına yazılır.  
/* Sadece açıklama olan satırlar */  
/*arasına yazılır.*/  
  
cout<<"merhaba"<<endl;  
//cout ekrana yazdırma komutudur. Ekrana yazdırılacak sayısal olmayan bilgiler "" arasında  
yazılır.  
//endl bir sonraki satıra yazdırır.  
  
system("pause");  
//Ekrana yazılan ifadenin görülmesi için system komutu kullanılarak akış durduruldu.  
//Programda ekranın durması sağlanır.  
  
return 0;  
  
/*C++ ve C'de her program en az bir main fonksiyon içerir. Fonksiyonlardan alınan değer ile  
geri dönüş komutu return'dür. Parametre dönmemiği için return 0 kullanılır.*/
```

- Açıklama satırları derleyici tarafından işlenmez. **Bu sebeple de tüm programlama dillerinde açıklama satırları yeşil renklidir.** İç içe yapıda açıklama satırı kullanılmaz. Buna dikkat edilmelidir. Örneğin aşağıdaki satır açıklama olarak yazıldığında ilk /* 'dan sonraki ifadeler açıklama olarak kabul edilir ve derleyici ilk */ görünce açıklama satırını sonlandırır. Aşağıdaki ifade Studio C++ editöründen alınmıştır. Siyah renkli kısım açıklama satırı dışında kalmıştır ve programda hataya yol açacaktır.

```
/* Sadece açıklama olan satırlar /*...*/ arasında yazılır.
```

- 3 ve 4. satırlar C++ kütüphanelerini eklemeye yarayan ön işlemci komutlarıdır. Ön işlemci C programları derlenmeden önce çalışan bir programdır. **include** komutu ön işlemci komutu olup ismi verilen kütüphanelerin içeriğini çalışılan programa dahil eder.
- C++ 'da ilgilendiğimiz ilk kütüphane iostream 'dir. C++ 'da giriş çıkış işlemleri C diline dahil değildir. Bunun yerine kütüphaneler düzenlenmiştir ve giriş/çıkış işlemleri

yapılacak ise programa <iostream> kütüphanesi dâhil edilir. cin (veri okuma) akışının gerçek tipi istream(input stream) ve cout(veri yazma) akışını gerçek tipi ostream(output stream)'dır. Diğer iki standart stream ise cerr ve clog'dur.

- Kütüphaneler C ve C++ için çok önemlidir. Kütüphaneler çeşitli işlemler için daha önceden yazılmış ve saklanmış fonksiyonları barındırırlar. Bizi daha önceden yazılan birçok fonksiyonu tekrardan yazma zahmetinden kurtarırlar. Bizde kütüphane oluşturabiliriz. Programımıza kütüphane eklemek için include komutu kullanırız. Kütüphanelerin uzantısı .h'dır.

Programlarımıza kullanabileceğimiz iki tip kütüphane vardır. C++ ile gelen ve bizim oluşturduğumuz. Eğer kütüphane tarafımızdan oluşturuldu ise;

```
#include "books.h"
```

şeklinde ekleriz. Eğer C++ kütüphanesi ise;

```
#include <iostream> şeklinde ekleriz.
```

- C++ bilgisayara çeşitli komutlar vererek çalışır. Bu komutlara atama gibi bakılabilir. Her atama bir fonksiyon olup, en temel fonksiyon main'dir. Bir C++ programı yazılıp çalıştırıldığında bakılacak ilk şey main fonksiyonunun olup olmadığıdır.
- satır using namespace std ise std olarak adlandırılan özel bir bölgede isimlendirilen nesnelerin kullanılacağını belirtir.
- 6. satır int main() {} işaretleri içerisindeki ana fonksiyondur. Tüm C++ programları main ana fonksiyonuna sahiptir. Eğer return 0 komutu kullanılmak istenmiyor ise void main() yazılır.
- 7. satır cout ekrana yazdırma komutudur. <iostream> kütüphanesinde bulunan bir komuttur. (C out yani C'nin çıkış komutudur.) C++ standart çıkış ortamı olarak ekranı kabul eder. << operatörleri ise yerleştirme operatöridür. Cout<< Ekrana yerleştir anlamındadır.

`cout<<"merhaba"` deyimi ekrana merhaba yazar.

`system ("pause")` ise bir tuşa basılana kadar programın çalışmasına ara verir.

`return 0` main ana fonksiyonundan dönen parametreyi içerir. Tüm fonksiyonlarda bir return komutu olmalıdır.

6.2. Değişken ve Veri Tipleri

6.2.1. Değişkenler

Değişken: Program içinde değeri değiştirebilen ifadedir.

Sabit: Program içinde değeri değişmeyen verileri saklayan, sadece tek bir değer alan ifadedir.

Operatör: Mantıksal veya matematiksel işlemler için kullanılan sembollerdir.

C++'de değişkenlere isim verirken aşağıdaki kurallara uyulmalıdır.

Değişken ismi;

1. Alt tire(an underscore “_”) veya harf ile başlar.

Doğru değişken isimleri: ad, soyad, _yas

Yanlış değişken isimleri: -name, adı soyadi, lpat

2. Harf, alttire veya sayıları içerir.

Doğru değişken isimleri: klavye, sayıl, tam_sayı, adı_soyadi

Yanlış değişken ismi: 1tam, Türkçe

3. Özel karakterler içermez.(!, %,], or \$)

4. Boşluk içermez.

5. Herhangi bir **rezerve kelime*** içermez.

6. C++ büyük küçük harf duyarlıdır. ISIM, isim, ISim, ISIml farklı değişken ismidir.

Rezerve Kelime C++'ın programlama komutları, değişken tanımlamaları v.s 'de kullandığı ingilizce kelimelerdir. Bu kelimeler aşağıda verilmiştir.

C++ Rezerve Kelimeleri				
Asm	Auto	bad_cast	bad_typeid	Bool
Break	Case	catch	Char	Class
Const	const_cast	continue	default	Delete
Do	Double	dynamic_cast	Else	Enum
Except	Explicit	extern	False	Finally
Float	For	friend	Goto	If
Inline	Int	long	Mutable	Namespace
New	Operator	private	Protected	Public

Register	reinterpret_cast	return	Short	Signed
Sizeof	Static	static_cast	Unsigned	Struct
Switch	Template	this	Throw	True
Try	type_info	typedef	Typeid	Typename
Union	Unsigned	using	Virtual	Void
Volatile	wchar_t	while		

6.3. C++'ta kullanılan veri tipleri

Programlar çeşitli veri türlerini işlerler. Bu veri tipleri sayısal veya alfayyal olabilir. Aşağıda C++ için veri tipleri verilmiştir.

Tam sayı veri tipi

Tam sayı değişkenleri üç tiptir. int, short ve long veri tipi kullanılır. İşletim sistemlerine göre hafızada kapladıkları yer değişmektedir. PC'ler için hafızada kapladıkları yer ve maximum sayı değerleri tabloda verildiği gibidir. C++'da tam sayı değişkeni tanımlamak ve ilk değer atamak için aşağıdaki formatlardan biri kullanılabilir.

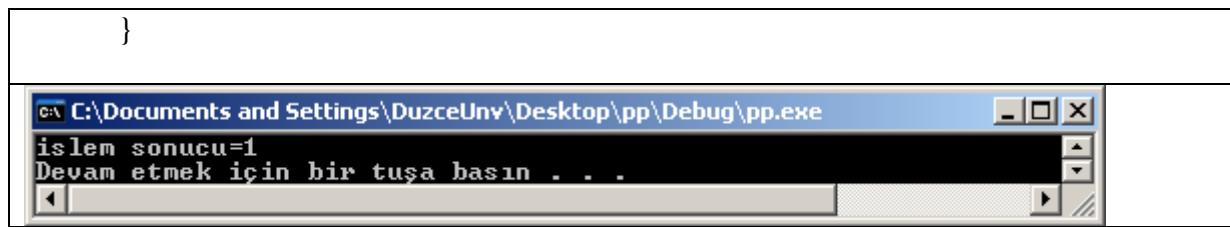
```
int a,b,c;//3 adet tamsayı degisken tanimalndi ilk deger atanmadı
int a,b,c=3;//3 adet tamsayı degisken tanimalndi c=3 ilk degeri atandi
int a=42,b,c=3;//3 adet tamsayı degisken tanimalndi a=42,c=3 ilk degeri atandi
int a(42),b,c(3);//3 adet tamsayı degisken tanimalndi a=42,c=3 ilk degeri atandi
int a(42);float b=34.45,c(3);//1 adet tamsayı degisken tanimalndi a=42, iki adet
float degisken tanimlandı b=34.45 ve c=3 ilk degeri atandi
```

Short	İnt	Long
2Byte	4 Byte	8 Byte

int türünde tamsayı, sabit veya ifadeler tamsayı sonuç üretirler.

Örnek: Aşağıdaki programın çalışması sonucu işlem sonucu ne olur?

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void _tmain(int argc, _TCHAR* argv[])
{ /* Bu bir açıklama//*Satırıdır*/
    int a=10,b=5,c=32;
    cout<<"islem sonunu="<<a*b/c<<endl;
    system("pause");
}
```



Floating Point Veri Tipi

Floating Point tip gerçek hayatı kullandığımız tam kısım ve ondalıklı kısımı olan sayıların gösteriminde kullanılır. Float, double ve long double olmak üzere üç farklı float veri tipi mevcuttur.

Float	Double	Long double
4Byte	8 Byte	8 Byte veya 16 Byte

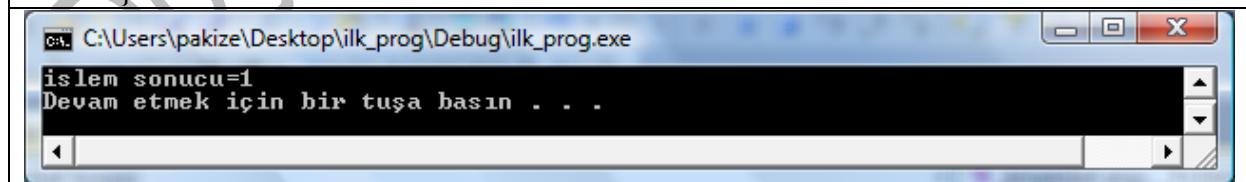
Float değişken tanımı;

```
float degisen_adi;
```

```
float a; şeklindedir.
```

Örnek: Aşağıdaki programın çalışması sonucu a değeri ne olur?

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void _tmain(int argc, _TCHAR* argv[])
{ /* Bu bir açıklama//Satırıdır*/
    int a,b=10,c=32;
    float x,y,z;
    x=4.55;
    a=x*b/c;
    cout<<"islem sonucu="<<a<<endl;
    system("pause");
}
```



Karakter Veri tipi

Bir karakter saklayabilen veri tipidir. C++ karakter veri tipi '' arasına yazılır ve işaretli(-128 ile 127) işaretsiz(0-255) arasında bir değere karşı düber. Karakter veri tipinde 64 @ sembolünü temsil eder. Karakter veri tipleri üzerinde aritmetik işlem yapılabilir. Örneğin

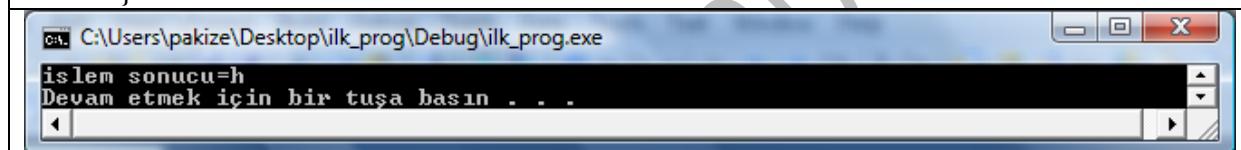
'c'+2→'e' yi temsil eder. Karakter Değişkenler, küçük harfler(a,b,c....z), büyük harfler(A,B...Z), sayılar(0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) ve özel karakterler(: ` ~ # \$! @ % ^ & * ({ [) }] | \ : ; “ ‘ + - < _ ? > , / = .)'i içerir.

C++da karakter değişken tanımlamak için ;

char degisken_adı ifadesi kullanılır.

Örnek: Aşağıdaki programın çalışması sonucu t değeri ne olur?

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void _tmain(int argc, _TCHAR* argv[])
{/* Bu bir açıklama//Satırıdır*/
    char t='c';
    t=t+5;
    cout<<"islem sonucu="<<t<<endl;
    system("pause");
}
```



String ve Karakterler

"Pakize" ifadesi string 'p' ifadesi ise karakter sabittir. ASCII kodlamadan dolayı bir karakter 1 byte'lık yer kaplar. "pakize" string sabiti ise hafızada 6 byte yer kaplar.

Karakter sabitler Char cevap='e', cinsiyet='b' şeklinde ‘’ arasında bir karakter ile tanımlanır.

Şu ana kadar char, int, long, float, double ile uğraştık. String nesneler ile uğraşmak için string kütüphanesi dahil edilir. String ifadeler “ifade” şeklinde tanımlanır.

```
#include <string>
```

String mesaj="merhaba" şeklinde atama yapılabilir.

mesaj=mesaj+"Arkadaşlar" şeklinde ekleme yapılabilir.

int uzunluk=mesaj.size() ile mesaj stringinin uzunluğu öğrenilebilir.

Örnek:

```
//String işlemleri
//17-02-2011 tarihinde Pakize ERDOGMUS tarafından yazıldı.
#include "stdafx.h"
#include <iostream>
#include <string>
```

```
using namespace std;
int
main(){
    int a;
    string mesaj="merhaba";
    mesaj=mesaj+"arkadaşlar";
    int uzunluk=mesaj.size();
    cout<<mesaj<<" mesajının uzunluğu="<<uzunluk<<"karakterdir";
    cin>>a;
    return 0;
}
```

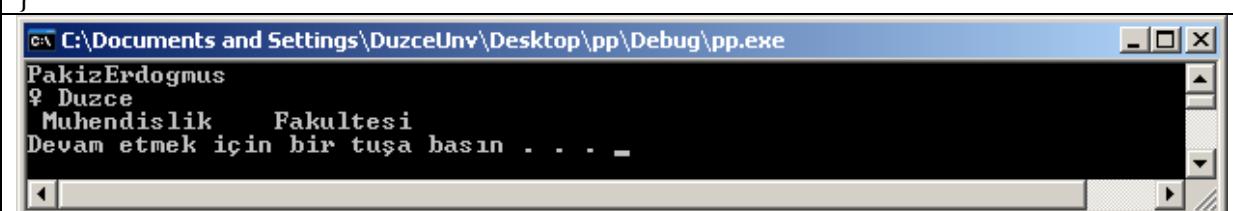


Tablo x. String ifadelerle kullanılan kaçış karakterleri

Kaçış Karakteri	Görevi
\n	Yeni Satır(Newline)
\t	Yatay Tab(Horizontal Tab)
\b	Boşluk(Backspace)
\a	Alarm(Alert bell)
\\\	Backslash
\”	Double quote
\v	Dikey Tab(Vertical Tab)
\f	Sayfa ilerletme(Form feed)
\r	Satır Başı(Carriage return)
\'	Single quote
?	Soru İşareti(Question mark)

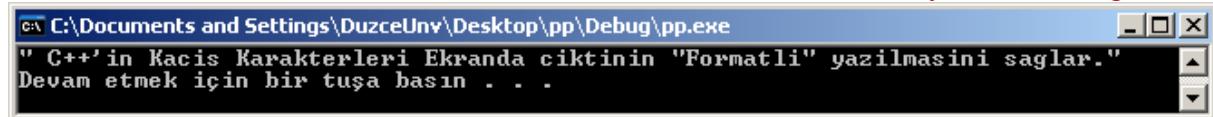
Örnek: Aşağıda kaçış karakterlerinin kullanımı ile ilgili bir örnek verilmiştir.

```
#include "stdafx.h"
#include <iostream>
int _tmain(int argc, _TCHAR* argv[])
{
    std::cout<<"Pakize\a"<<"\bErdogmus"<<std::endl;
    std::cout<<"\f Duzce\n"<<"Universitesi"<<"\r
Muhendislik"<<"\tFakultesi"<<std::endl;
    system("pause");
    return 0;
}
```



Örnek: Ekrana “C++’in Kacis Karakterleri Ekranda Ciktinin “Formatlı” yazilmasini saglar.” yazdırınız.

```
std::cout<<"\" C++'in Kacis Karakterleri Ekranda ciktinin \"Formatli\" yazilmasini saglar.\"";
```



Değişken tanımı

C++’da bir değişkeni kullanmadan önce o değişkene ait veri tipi tanımlanmalıdır. Örneğin yas değişkeni tanımlamak istiyorsanız, önce bu değişkende ne tür veri tipi kullanacağınızı belirtmeniz gereklidir.

int yas: yas değişkeni tamsayı veriler saklayacak.

char cinsiyet: cinsiyet değişkeni karakter veri saklayacak.

Sabit tanımı

Eğer programda değişimini istemediğiniz değerler var ise **const** ifadesi ile sabit tanımlayabilirsiniz.

Const tipi degisen_adı=degeri

Örnek:

```
const double t=3.14;
Const double pi=22.0/7;
```

Lokal ve Global Değişken tanımı

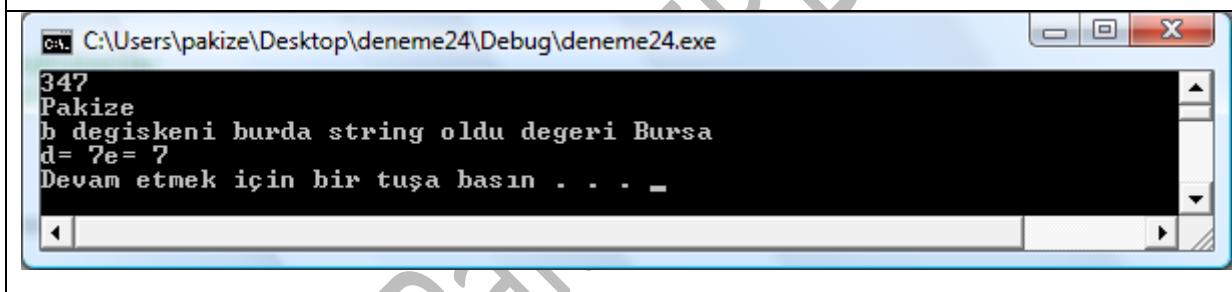
Her programlama dilinde olduğu gibi C’de de lokal ve global değişkenler tanımlanabilir. Bir değişken hiçbir {}(blok) iki küme işaretçi içinde yer almayırsa bu değişken global değişkendir ve tüm alt fonksiyonlar ve ana fonksiyonda tanımlıdır.

Lokal değişken ise hangi blok içinde ise etki alanı bu aralık olup bu aralığın dışında tanımlı değildir. Aşağıdaki örnek program ve çıktıları incelenerek olursa konu daha iyi anlaşılabilir.

Bir değişken global ve lokal değişken olarak farklı tiplerde tanımlanabilir. Örneğin önce global değişken olarak int ve lokal değişken olarak char olarak tanımlanabilir. **Eğer programın herhangi bir yerinde global olan tanıma ulaşılmak istenirse değişken ismi önüne :: işaretleri konulur.**

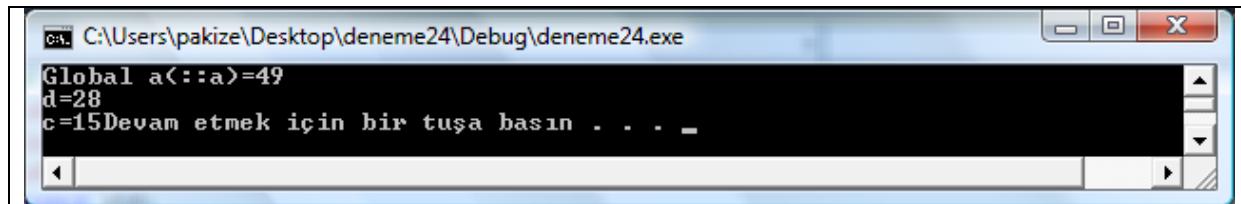
```
// string_class.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include <iostream>
```

```
#include <string> // file stream library
using namespace std;
int a,b,c;//Global değişkenler
string k;//Global değişken
int main() {
    a=3;b=4;c=a+b;
    k="Pakize";
    cout<<a<<b<<c<<endl;
    cout<<k<<endl;;
    {
        int d,e;//Lokal değişkenler
        d=a+b;
        e=c;
        string b="Bursa";//Lokal değişken
        cout<<"b değişkeni burda string oldu değeri "<<b<<endl;
        cout<<"d= " <<d<<"e= " <<e<<endl;
    }
    // burda cout<<d; yazılısa hata verir. Çünkü d değişkeni bu blokta tanımlı değildir.
    system("pause");
}
```



Örnek: Aşağıdaki programın çalışması sonucu lokal ve global değişken içerikleri ne olur?

```
// Lokal ve global değişkenler
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;
int a=4,b=3,c=12;
int
main(){
    int a=45,b=12,c=3;
    int d;
    d:::a+b+::c;
    c=c+::c;
    ::a:::a+a;
    cout<<"Global a(::a)=" <<::a<<endl<<"d=" <<d<<endl<<"c=" <<c;
    system("pause");
    return 0;
}
```

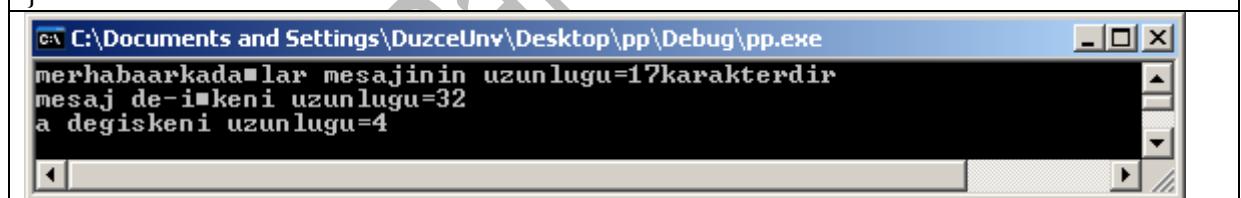


Herhangi bir veri tipinin bilgisayarda byte olarak ne kadar yer kapladığı **sizeof komutu** ile öğrenilebilir.

Örnek: Asağıdaki programda sizeof komutu ile değişken uzunluklarını görüntüleyiniz.

```
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;
int
main(){
    int a=45;
    string mesaj="merhaba";
    mesaj+= "arkadaşlar";
    int uzunluk=mesaj.size();
    cout<<mesaj<<" mesajının uzunluğu=" <<uzunluk<<" karakterdir" <<endl;

    cout<<"mesaj değişkeni uzunluğu=" <<sizeof(mesaj)<<endl<<"a değişkeni
uzunluğu=" <<sizeof(a);
    cin>>a;
    return 0;
}
```



C++’da değişkenlere başlangıç değeri verilmez ise ne olur?

C++ bir nesne kullanmadan önce tanımlanır.

```
int a
```

Veya

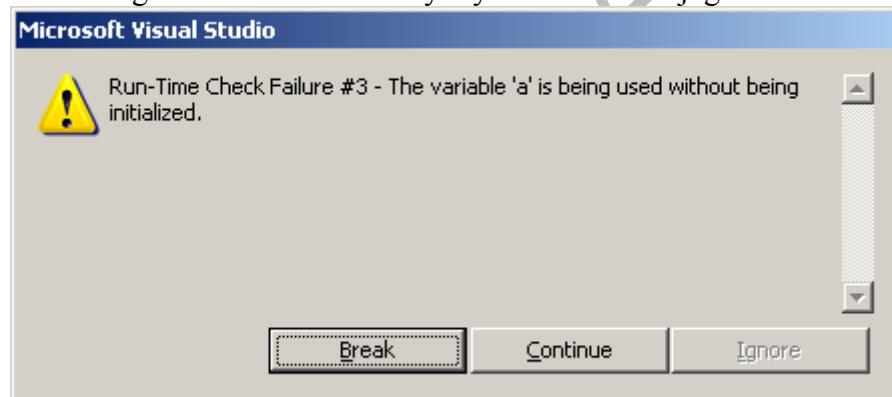
```
int a=3 şeklinde ilk değer vererek tanımlayabiliriz.
```

Örnek: İlk değer verilmeyen değişkenler ve içeriklerini görüntüleyen program.

```
ilk_degersiz_degiskenler
// ilk_degersiz_degiskenler.cpp : Defines the entry point for the console application.
//Bu program 17_02_2011 tarihinde Pakize ERDOGMUS tarafından Studio 2010'da yazıldı.
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int a;
    float f;
    char c;
    double d;
    cout<<"Tamsayı olarak tanımlanan a'nın değeri\n"<<a;
    cout<<"Float olarak tanımlanan f nin değeri"<<f<<endl;
    cout<<"Karakter olarak tanımlanan C değeri"<<c<<endl;
    cout<<"Double olarak tanımlanan d değeri"<<d<<endl;
    cin>>a;
    return 0;
}

C:\Documents and Settings\DUzceUniv\Desktop\cpp\ilk_degersiz_degiskenler\Debug\ilk...
Tamsayı olarak tanımlanan a'nın değeri
-858993460Float olarak tanımlanan f nin değeri-1.07374e+008
Karacter olarak tanımlanan C değeri!
Double olarak tanımlanan d değeri-9.25596e+061
```

Yukarıdaki program çalıştırıldığında önce "İlk değer atanmadı" hatası verecektir. Devam edilirse ekran çıktısı yukarıdaki gibi olacaktır. Aşağıda değişkenlerin ilk değer verilmeden kullanıldığı konusunda kullanıcıyı uyarın hata mesajı görülmektedir.

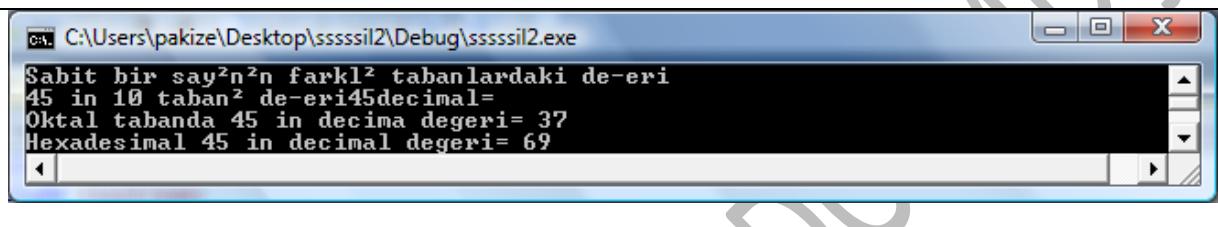


Farklı tabandaki sayıların C'de yazılışı

Decimal	önek yok
Hexadecimal	0X
Oktal	0
Örneğin 45 sayısının farklı tabanlardaki yeri:	
$45H = 4 * 16 + 5 * 1 = 69;$	
$45O = 4 * 8 + 5 = 37$	
$45D = 45$ bulunur.	

Örnek: Farklı tabanlardaki 45 sayısının 10 tabanına göre değerini veren program.

```
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;
void main()
{
    int a;
    cout<<"Sabit bir sayının farklı tabanlardaki değeri\n";
    cout<<"45 in 10 tabanı değeri"<<45<<"decimal= "<<endl;
    cout<<"Oktal tabanda 45 in decimal degeri= "<<045<<endl;
    cout<<"Hexadesimal 45 in decimal degeri= "<<0x45<<endl;
    cin>>a;
}
```



Üslü sayıların C++’da gösterimi;

Sayı.digit E üs şeklinde yazılır.

45×10^{23} C++’da $45e23$ şeklinde yazılır. $4.5E22$ şekline dönüşür.

C++’da NameSpace tanımı

Yukarıda ilk programlarımızı yazarken namespace’ı açıklamıştık. Tüm programlara da using namespace std eklemiştik. C++ programlama dilinde kullanılacak değişkenler ve fonksiyonlar çeşitli ad alanları içinde tanımlanırlar. Ad alanları programda farklı amaçlar için tanımlanarak kullanılan fonksiyon ve değişken isimlerinin karışmasını önerler.

Örneğin; aşağıdaki programda tanımlı iki toplam ve sayı değişkeni vardır. Her birine ait olana scope resolution sembolü ile ulaşırız.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
namespace sabitler{const double pi=22.0/7, eps=1.0/2e20, gravity_const=10;}
void main()
{
    cout<<"Cok kucuk sayi= "<<sabitler::eps;
    cout<<endl<<"Gravity sabiti ="<<sabitler::gravity_const;
    cout<<endl<<"Pi sayisi="<<sabitler::pi;
    system("pause");
}
```

Eğer namespace içinde çok sayıda değişken ve fonksiyon tanımlı ise her seferinde bu değişkenlere ulaşmak için scope resolution kullanmak yerine aşağıda görüldüğü gibi using namespace bildirim komutu kullanılır.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
namespace sabitler{const double pi=22.0/7, eps=1.0/2e20, gravity_const=10;}
void main()
{using namespace sabitler;
    cout<<"Cok kucuk sayi= "<<eps;
    cout<<endl<<"Gravity sabiti ="<<gravity_const;
    cout<<endl<<"Pi sayisi="<<pi;
    system("pause");
}
```

Ancak using namespace komutunu dikkatli kollanmak gereklidir. Örneğin her iki namespacede de aynı değişken var ise ve namespaceslerin herikisi içinde using namespace bildirimi yapılmışsa ambiguous iletisi alınır. Derleyici hangi namespace deki sayı değişkenini kullanacağını bilemez.

6.2. C++ Programlarında Yazım Kuralları

C++ yazım kuralları bakımından son derece kullanışlı bir dildir. Bu kurallardan bazıları aşağıda verilmiştir.

Komut sonu noktalı virgül ile bitmelidir.

C++ deyimleri kelimeyi bölmemek koşulu ile birden fazla satırda yazılabilir.

Örnek:

```
cout<<a<<A; veya
```

```
cout      <<a<<A; şeklinde yazılabilir. Fakat
cou      t<<a<<A; şeklinde yazılamaz.
```

Basit Giriş/Cıkış Biçimlendirme Formatları

C++’da konsoldan bilgi girişi için **cin**, konsola yazmak için **cout** deyim ikilisi kullanılır.

cout: Daha önceki konulardan bildiğiniz gibi cout deyimi konsola bilgi yazmakta kullanılır.

```
cout<<"String mesaj"<<degiseken_1<<degisken_2;
```

şeklinde kullanılır.

cin : Konsoldan bilgi okuma deyimi;

```
cin>>degisken_ismi
```

şeklinde kullanılır. Ancak kullanıcıya bir uyarı ve bilgi vermeden cin kullanılırsa kullanıcı ne gireceğini bilemez. Bu sebeple önce cout ile bilgilendirme yapmamız gereklidir.

Örneğin

```
cout<<"Yasınızı giriniz"<<endl;
```

cin>>yas; veya cin>>a>>b>>c şeklinde bir deyimla birden fazla değişken girilebilir. Artık C++’da basit algoritmaları kodlayabiliyoruz.

Örnek: Dairenin alanını bulan programın algoritması ve C++ kodu.

1. Basla
2. Dairenin yarıçapını gir.

3. Alan=pi*yarıçap*yarıçap
4. Ekrana yaz
5. Dur

C++ kodu

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    const double pi=22.0/7;
    double yarı_cap,alan;
    cout<<"Yarıçapı giriniz";cin>>yarı_cap;
    alan=pi*yarı_cap*yarı_cap;
    cout<<"Dairenin alanı=";<<alan;
    cin>>alan;
    return 0;
}
```

C++'da ilk değer atanırken;

i=0; j=0; ifadeleri yerine

i=j=0 yazılabilir.

Ifadeler

İfadeler bir operatör ile veya bir operatör işlemi olmadan bir değişkenden diğerine değer aktarmaya yarar. Örneğin;

a=5 bir aktarma ifadesi olup a değişkenine 5 değerini atar. Programlama dillerinde eşitlik matematik eşitliğinden farklı olup asıl görevi sağdaki değeri soldaki değişkene aktarmaktır.

a=5

a←5

a=a+5;

a←a+5=5+5=10 olur.

Örnek: Aşağıdaki atama ifadeleri sonucu değişkenlerin son değerleri ne olur?

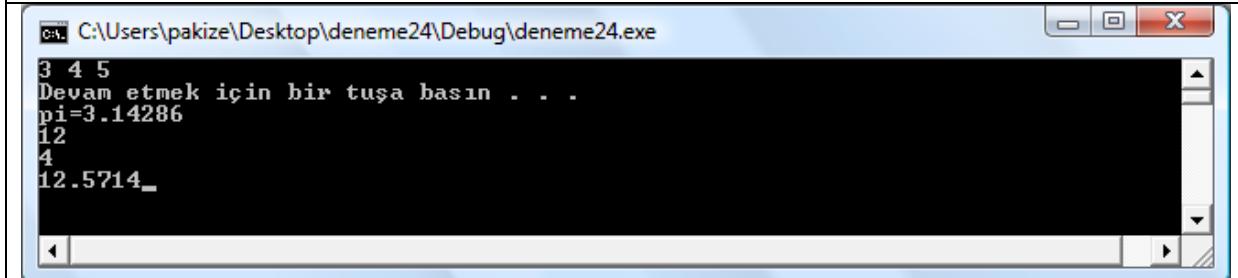
Örnek degiskenler

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    const double pi=22.0/7;
    int a,b;float c;
```

```

cin>>a>>b>>c;
a=c=pi*b;
system("pause");
cout<<"pi="<<pi<<"\n"<<a<<endl<<b<<endl<<c;
cin>>a;
    return 0;
}

```



6.4. Operatörler

Aritmetik operatörler, Mantıksal operatörler ve olmak üzere üç çeşit operatör mevcuttur.

Matematiksel İşlem Operatörleri

İşlem	Operatör	Örnek	Sonuç
Toplama	+	Top=3+3;	Top=6
Çıkarma	-	cik=3-2;	Cik=1
Çarpma	*	Carp=2*12;	Carp=24
Bölme	/	D=14/7;	D=2
Mod alma	%	S=13%4;	S=1
Negatif alma	-	a=3;b=-a;	b=-3
Son artım	++	T=2; A=T++;	A=2; T=3;
Ön artım	++	T=2; A=++T	T=3; A=3;
Son azaltım	--	T=3; B=T--;	T=2; B=3;
Ön azaltım	--	T=3;B=--T;	B=2; T=2;

Not: C++'da üs alma işlemi simbolu ^ değildir.(Matlab ve Basic'te üs alma simbolu ^'dır.)

t++ t'nin değerini bir arttırır. t++ ile t=t+1 aynı anlama gelemektedir. ++ operatörü değişken isminin başına veya sonuna eklenebilir. t++ ile ++t arasındaki fark vardır.

Örnek:

```
int t=2;
int p=3;
p++;//p= 4 oldu.
t=p++//t=4 oldu ve p=5 oldu.
t=++p//t=6 oldu ve p=6 oldu.
```

//t++ bir değişkene atandığında t'nin o andaki değeri değişkene atanır ve t sonra bir artar
 //++t 'de ise t önce bir artar sonra istenen değişkene t değeri aktarılır.

Atama Operatörleri

İşlem	Operatör	Örnek	Sonuç
Değer atama	=	a=3;b=a	b=3
Toplama Ataması	+=	a=a+3 a+=3	a=3+3=6
Çıkarma Ataması	-=	a=a-4 a-=4	a=6-4=2
Çarpma Ataması	*=	a=a*5 a*=5	a=2*5=10
Bölme Ataması	/=	a=a/3 a/=3	a=10/3=3
Mod Ataması	%=	a=a%1 a%=1	a=3%2=1

Karşılaştırma Operatörleri

İşlem	Operatör	Örnek	Sonuc
Eşit mi?	==	A=5, B=3 A==B	0
Eşit değil mi?	!=	A!=B	1
Küçük mü?	<	A<B	0

Büyük mü?	>	$A > B$	1
Eşit mi?	\geq	$A \geq B$	1
Küçük eşit mi?	\leq	$A \leq B$	0

Programlama Dilleri ve algoritmanın en önemli iki konusu atama ve karşılaştırmalardır. Bu iki konu tam anlaşılmaz ise programlama anlaşılamaz. Bu sebeple aşağıda tabloda bu iki kavram ile ilgili önemli noktalar verilmiştir.

ATAMA	KARŞILAŞTIRMA
<p>Atama bir sabit veya değişken değerinin başka bir değişkene aktarılmasıdır. (Yapılacak bir işi ifade eder. Bir komuttur.)</p> <p>$x=5$;(x değişkenine 5 sabit sayısı atandı.)</p> <p>$x=x+7$;(x değişkenine 7 sabit sayısı eklendi.)</p> <p>Sonuç olarak $x=12$;</p> <p>$y=2$;(y degiskenine 2 degeri atandı.)</p> <p>$x=x-y$;(x değişkeninden y değeri çıkarıldı.)</p> <p>(x artık 10)</p>	<p>Karşılaştırma ifadesi ise cevabı evet(bir)(doğru) veya hayır(sıdır)(yanlış) olan bir sorudur.</p> <p>$x=3$;</p> <p>$x==5$(x değişkeninin değeri 5'mi?)(Cevabı hayır,yanlış veya 0)</p> <p>$(x<4)$(Cevabı evet,doğru veya 1)</p>

ÖDEV: Aşağıdaki soruları soran karşılaştırma ifadelerini yazınız?

1. x değişkeni 10 ile bölünebilir mi?
2. x değişkeni tam sayı mı?
3. x değişkeni 56'dan büyük mü?
4. s sayısı 789'a eşit mi?
5. k sayısı 0'dan farklı mı?

Karşılaştırma Operatörleri Örnekleri

int a=5;

int b=8;

İşlem	Sonuç
a==b;//a b'ye eşit mi?	0
a<b; //a b den küçük mü?	1
a!=b;//a b den farklı mı?	1

Mantıksal Operatörler

İşlem	Operatör	Örnek	Sonuç
And(Ve)	&&	A>20 && A<30 A=26	1
Or(Veya)		A=34 A>=45 A<50	1
Not	!	A=30 C=!(A>20 && A<40)	1

NOT: Mantıksal OPERATÖRLER ile BİT DÜZEYİNDE OPERATÖRLERİ KARIŞTIRMAYINIZ

Bit Düzeyinde Operatörler

İşlem	Operatör	Örnek	Sonuç
Sola Öteleme(Shift left)	<<	32<<4(32'yi 4 kez sola ötele) 32=(0000100000) ₂ 64=(0001000000) ₂ 128=(0010000000) ₂ 256=(0100000000) ₂	512 Bu işlem 32×2^4 işlemine denktir.

		512=(10000000000)	
Sağ Öteleme(Shift right)	>>	32>>4	2 Bu işlemde $32/2^4$ işlemine denktir.
Bit Düzeyinde AND	&	4&8	0
Bit düzeyinde OR	 	4 8	12
Bit Düzeyinde XOR	^	4^8	12
Bir düzeyinde tamamlayan	~	~4	-5

Örnek: Aşağıdaki programının çalışması sonucu ifade sonuçları ne olur?

Mantıksal ifade örnekleri

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int m,n,p,q,r;
    m=45;n=6;p=15;q=24;r=10;
    cout<<(m&n)<<endl;
    cout<<(p|q)<<endl;
    cout<<(q^r)<<endl;
    cout<<~(n&p)<<endl;
    cout<<~(m>5)<<endl;
    cout<<(15<<3)<<endl;
    cout<<(m>30&&n<20)<<endl;
    cout<<(p>=15||q<5)<<endl;
    system("pause");
    return 0;
}
```

```
c:\Documents and Settings\... 4
31
18
-7
1
120
1
1
Devam etmek için bir tuşa basın . . .
```

$$\begin{array}{r}
 (\text{m}\&\text{n}) \\
 \text{m}=45=(0010\ 1101)_2 \text{ AND} \\
 \text{n}=6=(0000\ 0110)_2 \\
 \hline
 (0000\ 0100)_2=4
 \end{array}$$

(p|q)

$$\begin{array}{r}
 \text{p}=15=(0000\ 1111)_2 \text{ OR} \\
 \text{q}=24=(0001\ 1000)_2 \\
 \hline
 (0001\ 1111)_2=31
 \end{array}$$

$$\begin{array}{r}
 (\text{q}^{\wedge}\text{r}) \\
 \text{q}=24=(0001\ 1000)_2 \text{ XOR} \\
 \text{r}=10=(0000\ 1010)_2 \\
 \hline
 (0001\ 0010)_2=18
 \end{array}$$

$\sim(\text{n}\&\text{p})$

$$\begin{array}{r}
 \text{n}=6=(0000\ 0110)_2 \text{ AND}(\&) \\
 \text{p}=15=(0000\ 1111)_2 \\
 \hline
 =(0000\ 0110)=6 \text{ NOT}(\sim) \\
 (1111\ 1001)=-7 \text{ isbatı}
 \end{array}$$

Bir sayının negatifi ilk bire kadar(bir de dahil) aynı birden sonrası tersi idi. Yukarıdaki sayının bu şekilde negatifini alırsak +7 olduğu görülür.

$$(0000\ 0111)_2=7$$

$$\begin{array}{r}
 (\text{m}>>5)=\text{m}/2^5 \\
 \text{m}=45=(0010\ 1101)_2 \ll 5 \\
 \hline
 (0001\ 0110)_2 \\
 (0000\ 1011)_2 \\
 (0000\ 0101)_2 \\
 (0000\ 0010)_2 \\
 (0000\ 0001)_2=1 \text{ veya } 45/32=1
 \end{array}$$

$$\begin{array}{r}
 (15<<3)=\text{m}*2^3=15*8=120 \\
 15=(0000\ 1111)_2 \ll 3 \\
 \hline
 (0001\ 1110)_2 \\
 (0011\ 1100)_2 \\
 (0111\ 1000)_2=64+32+16+8=120 \\
 (\text{m}>30\&\&\text{n}<20) \\
 (45>30\&\&6<20)=(1\&\&1)=1
 \end{array}$$

$$\begin{array}{r}
 (\text{p}>=15\mid\mid\text{q}<5) \\
 (15>=15\mid\mid 24<5)=(1\mid\mid 0)=1
 \end{array}$$

Diger Operatörler

İşlem	Operatör
fonksiyon çağrıma	()
Dizi oluşturma	[]

Cast Operatörü

(Tip bildirisi)x şeklindedir. X değişkenini istenen tipe dönüştürür. Örneğin 32/15 sonucu 2 iken(Float) 32/15 sonucu 2.1333 eder. Veya fonksiyonlarda tip dönüşümünü sağlar.

Örneğin C++’da üs alma komutu pow(x,y)’dir. Ancak pow fonksiyonu double için tanımlıdır. Bu sebeple 2’nin küpünü almak için pow(2,(double)3); şeklinde yazılır.

Koşul Operatörü

Operatör	Sembülü	Örnek	Sonuç
Koşul operatörü	?	a?b:c	Eğer koşul doğru ise b ifadesi aksi halde c ifadesini çalıstır.

Örneğin a=3; b=2 olsun;

$a < b ? c = 1 : c = 0;$ ifadesinin anlamı şudur. Eğer $a < b$ ’den küçük ise $c = 1$ değilse $c = 0$ olsun. Verilen değerlere göre ($a < b$) karşılaştırma ifadesi sonucu yanlıştır. Yani $c = 0$ ifadesi çalıştırılır.

Örneğin 2. Dereceden denklem köklerini bulurken;

$\Delta \geq 0 ? \text{kok} = -b + \sqrt{\Delta} : \text{"Kök yok"}$

ÖRNEK: Aşağıdaki program delta sıfırdan büyük ve eşit iken kökleri bulmaktadır. Aksi halde sıfır değeri vermektedir.

ÖRNEK: Aşağıdaki koşul operatörü sonrası min değeri ne olur?

float diff, min, x1=100.0, x2=90.0, x3=10.0;

diff1=(x1>x2) ? (x1-x2) : (x2-x1);(x1 x2'den büyük olduğu için diff1=10 olur.)

$\min = (x_1 > x_2) ? ((x_2 > x_3) ? x_3 : x_2) : ((x_1 > x_3) ? x_3 : x_1)$ (x_1 x_2 'den x_2 'de x_3 'den büyük olduğu için min değeri x_3 olur.

ÖDEV: Yukardaki örnekte x_1, x_2, x_3 'e farklı değerler vererek **min** değerini gözleyiniz. Bu kodlar kodlardır. (Ne amaçla kullanılabilir.)

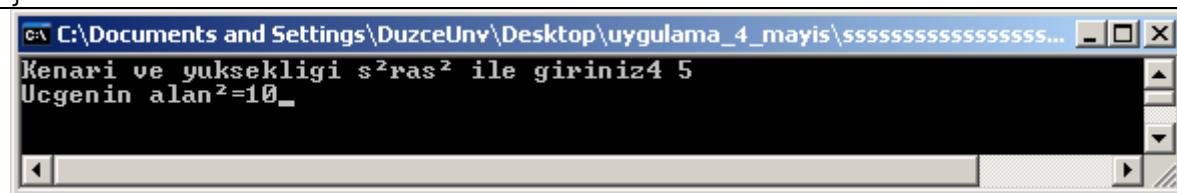
6.5. Basit Örnekler

Örnek: Klavyeden kenar uzunluğu ve yüksekliği girilen üçgenin alanını hesaplayan bir program yazınız.

```
Üçgen alani hesaplayan program örneği  
#include "stdafx.h"  
#include <iostream>
```

```
using namespace std;

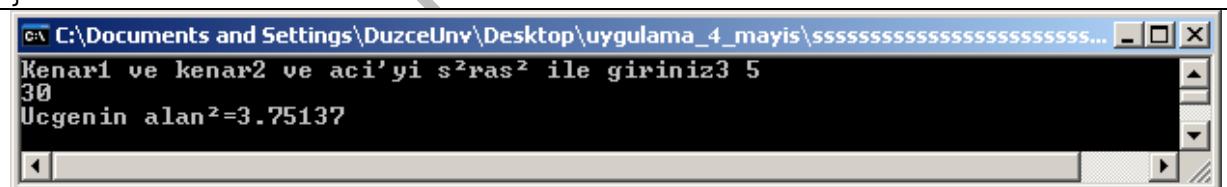
int _tmain(int argc, _TCHAR* argv[])
{
    float kenar, yukseklik, alan;
    cout<<"Kenari ve yuksekligi sirasi ile giriniz";cin>>kenar>>yukseklik;
    alan=kenar*yukseklik/2;
    cout<<"Ucgenin alani="<<alan;
    cin>>alan;
    return 0;
}
```



Örnek: İki kenarı ve aradaki açısı verilen üçgenin alanı hesaplayan bir program yazınız.

Üçgen alani hesaplayan program örneği

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    float kenar1,kenar2,aci,alan;
    const float pi=22.0/7;
    cout<<"Kenar1 ve kenar2 ve aci'yi(derece cinsinden) sirasi ile
giriniz";cin>>kenar1>>kenar2>>aci;
    alan=kenar1*kenar2*sin(pi*aci/180)/2;
    cout<<"Ucgenin alani="<<alan;
    cin>>alan;
    return 0;
}
```



Örnek: İki kenarı ve açısı verilen üçgenin üçüncü kenarını bulan algoritma ve C++ programını yazınız.

Algoritma:

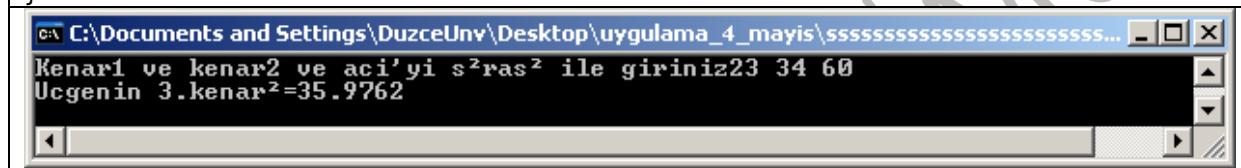
- 1. Basla**
 - 2. 1.kenar 2. Kenar ve açıyı gir.**
 - 3. $3.\text{kenar}^2 = (1.\text{kenar})^2 + (2.\text{kenar})^2 - 2 * (1.\text{kenar}) * (2.\text{kenar}) * \text{Cos}(aci)$**
 - 4. Yaz 3.kenar**
 - 5. Dur**

C++ Program kodu:

İki kenarı ve açısı verilen üçgenin üçüncü kenarını bulan algoritma ve C++ programınızı yazınız.

```
#include "stdafx.h"
#include <iostream>
#include <cmath>
using namespace std;

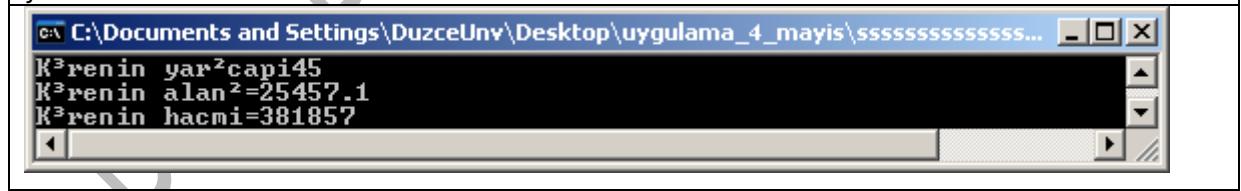
int _tmain(int argc, _TCHAR* argv[])
{
    float kenar1,kenar2,aci,kenar3;
    const float pi=22.0/7;
    cout<<"Kenar1 ve kenar2 ve acı'yi sırası ile giriniz";cin>>kenar1>>kenar2>>aci;
    kenar3=sqrt(kenar1*kenar1+kenar2*kenar2-2*kenar1*kenar2*cos(pi*aci/180)/2);
    cout<<"Ücgenin 3.kenarı="<<kenar3;
    cin>>kenar1;
    return 0;
}
```



Örnek: Kürenin alanı ve hacmini hesaplayan bir program yazınız.

Kürenin alanı ve hacmi

```
//Kürenin hacmi ve alanı
#include "stdafx.h"
#include <iostream>
using namespace std;
float r,V,A;
int main()
{
    cout <<"Kürenin yarıçapı";
    cin>>r;
    V=(4.0/3)*(22.0/7)*pow(r,3);
    A=4*(22.0/7)*pow(r,2);
    cout<<"Kürenin alanı="<<A<<endl;
    cout<<"Kürenin hacmi="<<V<<endl;
    cin>>r;
    return 0;
}
```



BÖLÜM SONU SORULAR VE UYGULAMALAR

```
#include "stdafx.h"
#include <iostream>
#include <cmath>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int i, float j,k;
    const float pi=22.0/7;
    j=i=k=pi;
    cout<<"i,j,k değerleri \n sırası ile\r verildiği gibidir.";
    cout<<"i="<<i<<"\nj="<<j<<"\k="<<k;
    system("pause");
    return 0;
}
```

1. Yukarıdaki kodun çalışması sonucu ekran çıktısı ne olur?

2.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
int i;
cout>>"merhaba">>endl;
system("pause");}
    return 0;
```

Programının hatalarını bulunuz.

3.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{ int i;
i=13/3*4;
cout<<i;
system("pause");
    return 0;
}
```

programın çalışması sonucu ekran çıktısı ne olur?

4.

```
#include "stdafx.h"
```

```
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{ int m,n,p,q,r;
m=4;n=2;p=10;q=8;
cout<<"m++="<<m++<<" " <<"m="<<m<<endl;
cout<<"--n="<<--n<<" " <<"n="<<n<<endl;
cout<<"-+p="<<-+p<<" " <<"p="<<p<<endl;
cout<<"-q-= "<<-q-<<" " <<"q="<<q<<endl;
r=m+n+p+q;
cout<<r<<endl;
m=-m++;
n=n;
p=p;
q=-++q;
cout<<"m=.."<<m<<endl;
cout<<"n=.."<<n<<endl;
cout<<"p=.."<<p<<endl;
cout<<"q=.."<<q<<endl;
system("pause");
return 0;
}
```

Programın çalışması sonucu değişken değerleri ne olur?

5. Dışarıdan girilen değişkenin $23 < x < 33$ olması durumunda 1 veren mantıksal ifadeyi yazınız.

6. Dışarıdan girilen x ve y değişkenlerinden herhangi birinin 50'den küçük olması durumunda 0 veren ifadeyi yazınız.

7. Değişkenin 20'den küçük veya eşit ve 80'den büyük veya eşit olması durumunda 1 veren mantıksal ifadeyi yazınız.

```
8. #include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{ int m,n,p,q,r;
m=14;n=18;p=9;q=2;r=10;
cout<<(m&n)<<endl;
cout<<(p|q)<<endl;
cout<<(q^r)<<endl;
cout<<(~(n&p))<<endl;
cout<<(m>5)<<endl;
cout<<(10<<3)<<endl;
cout<<(m>40&&n<50)<<endl;
cout<<(p>=25 || q<15)<<endl;
cout<<(float) 42/15<<endl;
cout<<pow(3,(double)3);
cout<<sqrt((double)45);
system("pause");
return 0;
}
```

Yukarıdaki programın çalışması sonucu işlem sonuçları ve ekran çıktısı ne olur?

7. C++'da Kontrol Ve Döngü Deyimleri

Her programlama dilinde mevcut olduğu gibi C++ 'da kontrol deyimleri mevcuttur. Bu kontrol deyimlerinin kullanımı C dilindekilerle aynıdır.

Kontrol deyimleri programın akışını değiştiren, yönünü belirleyen deyimlerdir. Kontrol deyimleri aynı zamanda döngü deyimleri ile iç içe kullanılabilir. Her bir döngü deyimi içerisinde döngünün ne zaman biteceğine dâhil kontrolun yapıldığı bir bölüm bulunmaktadır

Deyimler

Belli başlı kontrol ve döngü deyimleri aşağıdakilerdir.

if

if-else

switch - case

for

while, do-while

goto

Bu deyimlerle kullanılan bazı özel deyimler de şunlardır

break

continue

7.1. if-else

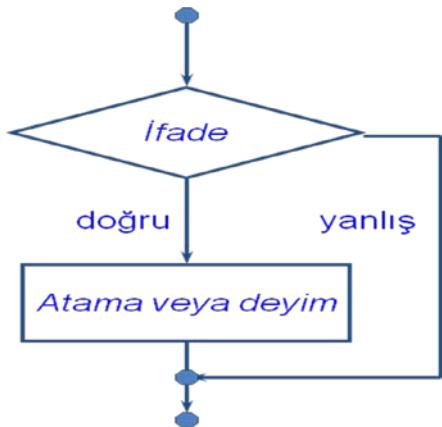
if deyimi parametre olarak aldığı değer doğru ise kendisinden sonra gelen fonksiyonu yada fonksiyonları gerçekleştirir. Eğer birden fazla deyim kullanılacak ise {} blok içersine alınır. Blok içine alma { } kıvırcık parantez arasına alma anlamına gelmektedir.

if'in kullanımı :

if ([ifade]) [deyim1];

deyim2

Eğer ifade doğru ise deyim1 ve deyim2 çalışır. Yanlış ise deyim1 çalışmaz, sadece deyim2 çalışır. Bu kullanım şekli if 'ten sonra sadece bir adet deyim çalıştırırmak içindir. İf akış şeması aşağıda verilmiştir.



Eğer birçok deyim çalıştırırmak istiyorsanız aşağıdaki gibi bir kod bloğu açmalısınız

```

if ( [fade] )
{
  [deyim];
  [deyim];
  ...
}
  
```

if kelimesinin Türkçe karşılığı eğer anlamına gelmektedir. Eğer if ile verilen ifade doğruysa if'ten sonraki bloktaki fonksiyonları gerçekleştir. Doğru değilse if ten sonraki bloğu atla (yok say).

if (fade) deyim1; else deyim2;

deyim3;

kodunda ifade doğru ise deyim1 ve deyim3 yanlış ise deyim2 ve deyim3 işlenir.

Dikkat edilecek nokta if deyiminden sonra ; (noktalı virgül) konulmamaktadır.

if-else'nin kullanımı :

```

if ( [fade] )
  [deyim];
else [deyim];
  
```

ya da

```
if ( [ifade] )
```

```
{
```

```
[deyim 1];
```

```
[deyim 2];
```

```
[deyim n];
```

```
}
```

```
else
```

```
{
```

```
[deyim 1];
```

```
[deyim 2];
```

```
[deyim n];
```

```
}
```

Örnek: Aşağıdaki algoritma ile girilen iki sayıdan büyük olan bulunuyor.

```
int a=4,b=43,c=7;
if (a<b) a=a+5; else c+=a+c;
b=b*a+c;
```

komutları sonrası a değişkeni b'den küçük olduğu için a=a+5 çalışır. a=4+5=9 olur.

b=43*9+7=387+7=394 bulunur.

Örnek: Aşağıdaki algoritma ile girilen iki sayıdan büyük olan bulunuyor.

İki sayıdan büyük olanı bulan program
<pre>#include "stdafx.h" #include <iostream> using namespace std; int _tmain(int argc, _TCHAR* argv[]) {int a,b,k=0,eb; cout<<"İki tam sayı giriniz?"<<endl; cin>>a>>b; if (a<b) eb=b; else if (a>b) eb=a; else k=1; if (k==1) cout<<"Sayilar birbirine esit"<<endl; else cout<<"En büyük sayi="<<eb<<endl; system("pause"); return 0; }</pre>

```
C:\Documents and Settings\Du\u011fzceUniv\Desktop\uygulama_4_mayis\ssssssssssssssssssssss...  
!iki tam sayı giriniz?  
454 7987  
En büyük sayı=7987  
Devam etmek için bir tuşa basın . . .  
!iki tam sayı giriniz?  
4564 4564  
Sayılar birbirine eşit  
Devam etmek için bir tuşa basın . . .
```

Örnek: İkinci dereceden denklem köklerini bulan programı yazınız.

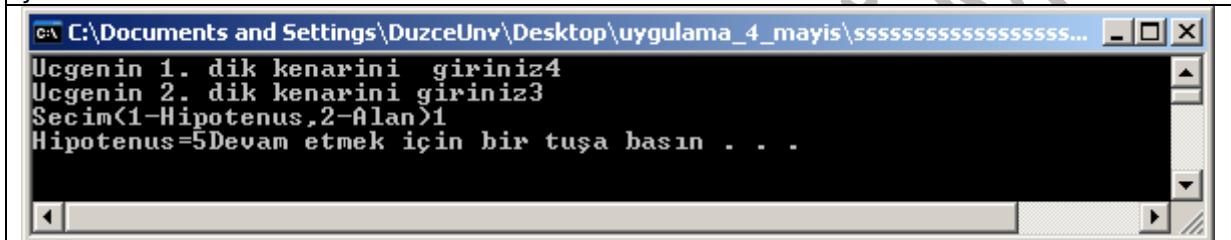
Örnek: Dışarıdan girilen seçime göre bir dik üçgenin alanı veya hipotenüsünü hesaplayan programı yazınız.

```
Bir dik üçgenin alanı veya hipotenüsünü hesaplayan program  
#include "stdafx.h"  
#include <iostream>  
using namespace std;  
float h,secim;  
float a,b,alan;  
int main()  
{
```

```

cout<< "Ucgenin 1. dik kenarini giriniz";
cin>>a;
cout<< "Ucgenin 2. dik kenarini giriniz";
cin>>b;
secim: cout<<"Secim(1-Hipotenus,2-Alan)";
cin>>secim;
    if (secim!=1 && secim!=2)
    {
        cout<<"Secim 1 veya 2 olabilir";
        goto secim;
    }
    if (secim==1)
        {h=sqrt(a*a+b*b);
        cout<<"Hipotenus="<<h;}
    else
        {alan=a*b/2;
        cout<<"Alan="<<alan<<endl;}
    system("pause");
    return 0;
}

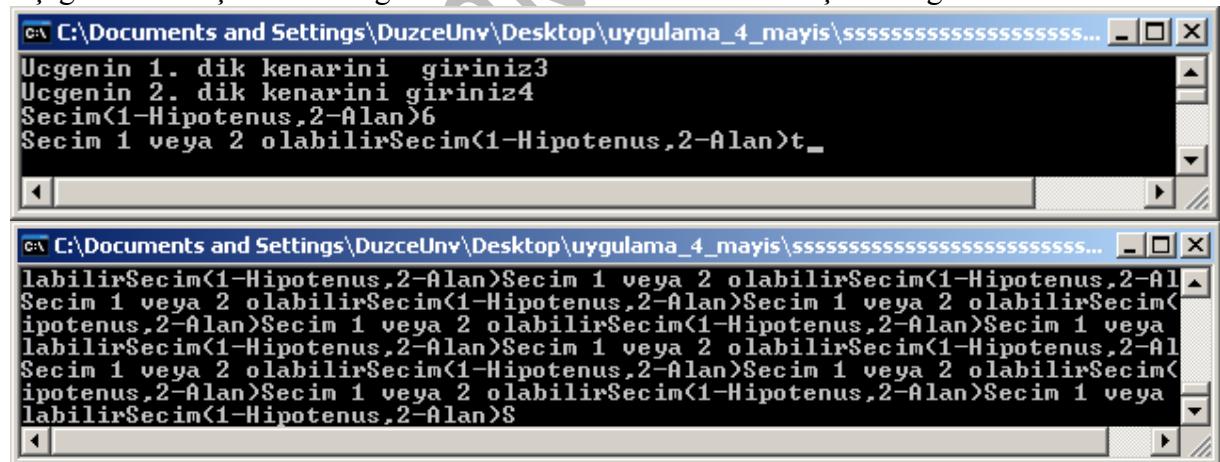
```



Yukarıdaki programı kenar değerlerine sıfır girilene kadar istenilen sayıda hipotenüs veya alan bulacak şekilde if ve goto kullanarak değiştiriniz.

Bu programda seçim sayısal bir değer olduğu için, dışarıdan sayısal olmayan bir değer girilmesi durumunda sonsuz döngüye girer. Bunu önlemek için char veri tipi ile alıp kontrol etmek daha kolay olur.

Asağıda seçim için karakter girilmesi durumu ve sonrası ekran çıktıları görülmektedir.



```
Bir dik üçgenin alanı veya hipotenüsünü hesaplayan program  
#include "stdafx.h"  
#include <iostream>  
using namespace std;  
float h;  
char secim;  
float a,b,alan;  
int main()  
{  
    cout<< "Ucgenin 1. dik kenarini giriniz";
```

```

cin>>a;
cout<< "Ucgenin 2. dik kenarini giriniz";
cin>>b;
secim: cout<<"Secim(1-Hipotenus,2-Alan)";
cin>>secim;
if (secim!='1' && secim!='2')
{    cout<<"Secim 1 veya 2 olabilir";
goto secim;}
if (secim=='1')
{h=sqrt(a*a+b*b);
cout<<"Hipotenus="<<h;}
else
{alan=a*b/2;
cout<<"Alan="<<alan<<endl;}
system("pause");
return 0;
}

```

```

C:\Documents and Settings\DUzceUnv\Desktop\uygulama_4_mayis\ssssssssssssssssssss... -> x
Ucgenin 1. dik kenarini giriniz3
Ucgenin 2. dik kenarini giriniz4
Secim(1-Hipotenus,2-Alan)>t
Secim 1 veya 2 olabilirSecim(1-Hipotenus,2-Alan)>3
Secim 1 veya 2 olabilirSecim(1-Hipotenus,2-Alan)>2
Alan=6
Devam etmek için bir tuşa basın . . . -

```

Örnek: Dışarıdan girilen bir notu okuduktan sonra aşağıdaki not değerlendirme sistemine göre sınıflayan bir program yazınız.

$$\text{Harf not} = \begin{cases} \text{not} < 45, \text{FF} \\ 45 \leq \text{not} < 55, \text{FD} \\ 55 \leq \text{not} < 60, \text{DD} \\ 60 \leq \text{not} < 70, \text{DC} \\ 70 \leq \text{not} < 85, \text{CB} \\ \text{not} \geq 85, \text{BA} \end{cases}$$

Not sınıflandırma programı

```

Not sınıflandırma programı
#include "stdafx.h"
#include <iostream>
using namespace std;
int not;
int main()
{
    cout<<"notu giriniz";cin>>not;
    if (not<45) cout<<"Notun harf karsiligi FF";
    else if (not<55) cout<<"Notun harf karsiligi FD";
    else if (not<60) cout<<"Notun harf karsiligi DD";
    else if(not<70) cout<<"Notun harf karsiligi DC";
    else if(not<85) cout<<"Notun harf karsiligi CB";
    else cout<<"Notun harf karsiligi BA\n";
    system("pause");
}

```

```

        return 0;
}

cv C:\Documents and Settings\DUzceUnv\Desktop\uygulama_4_mayis\ssssss... □ X
notu giriniz67
Notun harf karsiligi DC
Devam etmek için bir tuşa basin . . .

```

7.2. Blok

Blok C++’da {} arası demektir. Kontrol komutlarında(if, switch/case, for, do, while v.s) birden fazla deyim ve ifade yazılacağı durumlarda blok kullanıldığı gibi, doğrudan blok kullanılabilir. Bu durumda değişken tanımlamaları sadece ait oldukları blok içerisinde geçerlidir. Aşağıdaki örnek blok kullanımını göstermektedir.

Örnek: Blok ile değişkenlerin erişimlerinin değişimini inceleyiniz.

```

Blok tanimlari
#include "stdafx.h"
#include <iostream>
using namespace std;
int main ()
{//Ana fonksiyon blogu
    int x = 21;
{ //Birinci Blok ici
    int x =124,y=23;
{ //İkinci blok içi
    int x=98,y=67;
    cout<< "İkinci Blok ici degisen degerleri\n";
    cout<< "x="<<x<<endl;
    cout<< "y="<<y<<endl;
}
cout<<"Birinci Blok ici degisen degerleri\n";
cout<<"x="<<x<<endl;
cout<<"y="<<y<<endl;
}
cout<<"Birinci blogun disi\n"<<endl;
cout<<"x="<<x<<endl;
// İlegal artık y tanımlı değil cout<<"y="<<y;
system("Pause");
return 0;
}

cv C:\Documents and Settings\DUzceUnv\Desktop\uygulama_4_mayis\ssssssssssssssssssssss... □ X
İkinci Blok ici degisen degerleri
x=98
y=67
Birinci Blok ici degisen degerleri
x=124
y=23
Birinci blogun disi

x=21
Devam etmek için bir tuşa basin . . .

```

7.3. switch-case

Son örnekte görüldüğü gibi eğer kontrol edilmesi gereken işlem sayısı çok fazla ise if,elseif ve else deyimlerini takip etmek zorlaşmaktadır. İşte switch/ case bir değişkenin çeşitli değerlerine göre program kontrolünü yönlendiren bir ifadedir. Kullanımı aşağıda verilmiştir.

```
switch (değişken)
{
    case deger1:
    .....
    break;
    case deger2:
    .....
    break;
    .....
    case deger_n:
    .....
    break;
    default:
    .....
    break;}
```

şeklinde kullanılır. *Eğer girilen değişken değeri case ifadelerinde yer alan değerlerden biri değil ise bu durumda defaulttan sonraki deyimler işlenir.*

Örnek: İki kenarı girilen bir dik üçgende çeşitli hesaplamalar aynı programda yapılmak istensin. 4 farklı seçenek olsun.

1. Hipotenüs hesabı
2. Alan hesabı
3. Üçgenin açıları hesabı
4. Çıkış olsun.

Bu programı yazınız.

Çok seçenekli program örneği

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main() {
    int secim;
    float a,b,h,alan,teta1,teta2;
    cout<<"Bu program bir dik ucgenin hipotenusu, alani veya acilarini
hesaplar.\n\n"<<"Dik ucgenin kenarlarini giriniz";
    cin>>a>>b;
    basla:
        cout<<"\n1.Hipotenus\n2.Alan\n3.Acilar \n4.Cikis ";
        cout<<"\n\n Seciminizi giriniz\n\n";
```

```
    cin>>secim;
    switch (secim)
{case 1:
    cout<<"\n\tHipotenus="<<sqrt(a*a+b*b)<<endl;
    break ;
case 2:
    cout<<"\n\tAlan="<<a*b/2<<endl;
    break;
case 3:
    cout<<"\n\tBirinci aci teta1="<<atan(a/b)*180/((double)
22/7)<<endl<<"\n\tIkinci aci teta2="<<atan(b/a)*180/((double) 22/7)<<endl;
    break;
case 4:
    goto son;
    break;
default:
    cout<<"\nSeciminiz 1 ile 4 arasında bir tam sayı olmalıdır\n\n";
}
    goto basla;
son:   system("pause");
return 0;
}
```

```

C:\Documents and Settings\DUZCEUNV\Desktop\uygulama_4_mayis\ssssssssssssssssssssssssssss\De...
Bu program bir dik üçgenin hipotenüsü, alanı veya açılarını hesaplar.

Dik üçgenin kenarlarını giriniz4
3

1.Hipotenüs
2.Alan
3.Açılar
4.Cıkış

Seçiminizi giriniz

1
Hipotenüs=5

1.Hipotenüs
2.Alan
3.Açılar
4.Cıkış

Seçiminizi giriniz

2
Alan=6

1.Hipotenüs
2.Alan
3.Açılar
4.Cıkış

Seçiminizi giriniz

3
Birinci açı  $\theta_1 = 53.1087$ 

İkinci açı  $\theta_2 = 36.8551$ 

1.Hipotenüs
2.Alan
3.Açılar
4.Cıkış

Seçiminizi giriniz

4
Devam etmek için bir tuşa basın . . .

```

Örnek: Birden fazla case ifadesine tek bir atama yapılabilir. Örneğin dışarıdan girilen bir harfin sesli veya sessiz olduğunu bulan bir program yazınız.

Switch/case örneği

```

// This programme was developed for the SWITCH/CASE application on November the 14th
2011.
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{char c;
cout<<"Bu program girilen bir karakterin sesli harf veya sessiz harf olduğunu
bulur\n";
cout<<"Bir harf giriniz"<<endl;
cin>>c;
switch (c) {
    case 'a': case 'A':
    case 'e': case 'E':

```

```

case 'i': case 'I':
case 'o': case 'O':
case 'u': case 'U':
    cout << c << " sesli bir harftir.\n" << endl;
    break;
default:
    cout << c << " sessiz bir harftir" << endl;
}
system("pause");
return 0;
}

C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
Bu program girilen bir karakterin sesli harf veya sessiz harf olduunu bulur
Bir harf giriniz
p
p sessiz bir harftir
Devam etmek için bir tuşa basın . . .

```

Bu programın bir dezavantajı sesli harf dışında girilen tüm karakterleri sessiz harf olarak sınıflandıracaktır.

```

C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
Bu program girilen bir karakterin sesli harf veya sessiz harf olduunu bulur
Bir harf giriniz
a
a sesli bir harftir.

Devam etmek için bir tuşa basın . . .

```



```

C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
Bu program girilen bir karakterin sesli harf veya sessiz harf olduunu bulur
Bir harf giriniz
5
5 sessiz bir harftir
Devam etmek için bir tuşa basın . . .

```

Bunu düzeltmek için karakterleri ascii kod tablosu değerlerinden yararlanılabilir.

Büyük Harflerin ASCII kodları:65(A)-90(Z) arası

Küçük Harflerin ASCII kodları:97(a)-122(z) arası

Aşağıda aynı programın biraz daha gelişmiş bir versiyonu verilmiştir.

Switch/case örneği

```

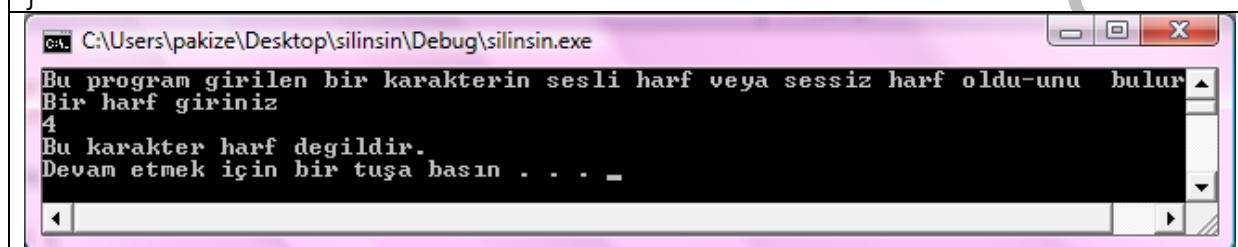
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{char c;
cout<<"Bu program girilen bir karakterin sesli harf veya sessiz harf olduğunu
bulur\n";
cout<<"Bir harf giriniz"<<endl;
cin>>c;
if ((c>64&&c<91)|| (c>96&&c<123))
{
    switch (c) {
        case 'a': case 'A':
        case 'e': case 'E':

```

```

        case 'i': case 'I':
        case 'o': case 'O':
        case 'u': case 'U':
            cout << c << " sesli bir harftir.\n" << endl;
            break;
        default:
            cout << c << " sessiz bir harftir" << endl;
    }
}
else
    cout<<"Bu karakter harf degildir."<<endl;
system("pause");
return 0;
}

```

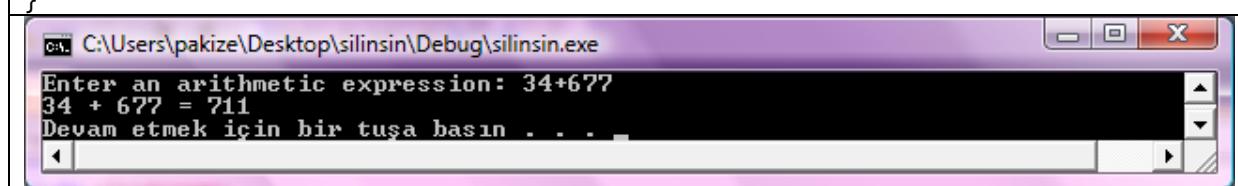


Örnek: Switch/Case ile dört işlem yapan bir program yazınız. Dışarıdan 1. Sayı operatör ve 2. Sayı girildiğinde operatöre göre işlem yapılacak.

```

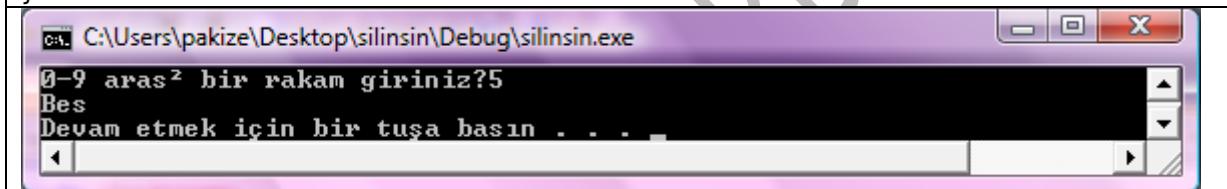
Switch case ile basit hesap makinesi
// This programme was developed for the SWITCH/CASE application on November the 14th
2011.
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{cout << "Enter an arithmetic expression: ";
int number1;
int number2;
char Operator;
cin >> number1 >> Operator >> number2;
cout << number1 << " " << Operator << " " << number2
<< " = ";
switch (Operator) {
    case '+': cout << number1 +number2 << endl; break;
    case '-': cout << number1 - number2 << endl; break;
    case '*': cout << number1 * number2 << endl; break;
    case '/': cout << number1 / number2 << endl; break;
    case '%': cout << number1 % number2 << endl; break;
    default: cout << "Wrong arithmetic operator was entered" << endl;
}
system("pause");
return 0;
}

```



Örnek: Tek basamaklı sayıların okunuşunu veren bir programı switch case ile yazınız.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int a;
    basla:
    cout<<"0-9 arası bir rakam giriniz?";
    cin>>a;
    switch (a)
    {
        case 0:cout<<"Sifir";break;
        case 1: cout<<"Bir";break;
        case 2:cout<<"İki";break;
        case 3:cout<<"Üc";break;
        case 4:cout<<"Dort";break;
        case 5:cout<<"Bes";break;
        case 6:cout<<"Alti";break;
        case 7:cout<<"Yedi";break;
        case 8:cout<<"Sekiz";break;
        case 9:cout<<"Dokuz";break;
        default: cout<<"Girdiginiz sayı rakam degil"; goto basla; break;
    }
    cout<<endl;system("pause");
    return 0;
}
```



Ödev: Switch ile iki basamaklı sayıların okunuşunu bulan bir program yazınız.

7.4. FOR

Bilgisayarda n kere tekrar etmesi gereken işlemler için kullanılan deyimlere döngü deyimleri denir. FOR kontrollü döngü deyimidir. Yani for ile kaç defalik bir döngü oluşturulacağı bellidir. Örneğin 1'den 5'e kadar say toplamını nasıl yaptığımızı düşünelim. Önce 1 ile başlıyoruz. Daha sonra 2 ekleriz toplam 3 ve bu toplama 3 ekleriz toplam 6 ve 4 ekleriz toplam 10 ve 5 eklediğimizde 15 olur. Bu şekilde elde ettiğimiz son toplam 1'den 5' e sayılar toplamıdır. Bilgisayar programları insanın problem çözme yeteneğini simule eder. Dolayısıyla for döngüsü yukarıdaki toplama işlemine benzer işlemlerde değişkeni başlangıç değerinden(1) son değerine(5) kadar artış miktarı(+1) kadar arttırarak n kez yapılması gereken işleri yapmamıza imkân sağlar. Kullanımı;

for (i=ilk durum; şartı;güncelleme)

{

İşlenecek deyimler

}

veya *for içerisinde işlenmesi gereken tek bir deyim var ise;*

for (i=ilk durum; şartı;güncelleme)deyim;

Burada deyim olarak ifade edilen şey bir atama veya tek bir C++ komutudur.

Sırası ile *ilk değer ataması, döngünün devam etme şartı ve artış miktarı ifadesi* girilir. Derleyici ilk değeri alır. Bu ilk değer devam etme şartını sağlıyor ise blok içerisindeki deyimler veya for'dan sonraki deyim işlenir. Sonra başa dönülür ve değişken ilk değeri günceller. Güncel değişken devam şartını sağlıyor ise tekrar döngü içi veya for deyimi işlenir. **Bu durum devam şartı yanlış olana kadar devam eder.**

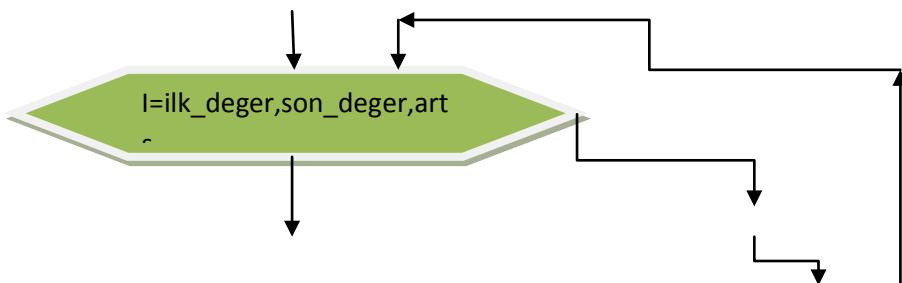
Örnek: 5'ten 1'e kadar sayıların çarpımını yapan bir programda kullanılan for döngüsü nasıl olmalıdır?

Güncellemeye i değerinin for ile verilen deyim işlendiğinden sonra bir azalacağı söylüyor. -- i olsaydı i ve faktörleri ne olurdu acaba?

```
int fakt=1,i;
for(i=5;i>=1;i--)
fakt=fakt*i;
```

i değeri	devam şartı	İşlem
i=5;	5>=1 Doğru	fakt=1*5;
i=4;	4>=1 Doğru	fakt=5*4;
i=3;	3>=1 Doğru	fakt=5*4*3;
i=4;	2>=1 Doğru	fakt=5*4*3*2;
i=1;	1>=1 Doğru	fakt=5*4*3*2*1;
i=0;	0>=1 Yanlış Döngüden çıkar.	

Akış şeması aşağıdaki gibidir.



ÖRNEKLER:

For (i=1;i<10;i++){} → Döngüsü 1'den 9'a kadar 1'er artan i sayıları oluşturur.

1,2,3,...9. Kıvırcık parantez içi 9 kez çalıştırılır.

For (i=0;i<=100;i=i+5){} → Döngüsü 0'dan 100'e kadar 5'er artan i sayıları oluşturur.

0,5,10,...100. Kıvırcık parantez içi $(100-0)/5+1=21$ kez çalışır.

For(i=50;i<20;i--) → Döngüsü hiç çalışmaz. Çünkü başlangıç değer 50 sonlandırma şartını sağlamıyor.

For(i=50;i>20;i--) → Döngüsü 50'den 21'e 1'er azalan i sayıları oluşturur. 50,49,48,...21.

Kıvırcık parantez için $(50-21)/1+1=30$ kez çalışır.

For(;;){int a;cout<<"sonsuz döngü çıkmak için bir rakamına bas"; cin>>a; if (a==1) break;}

For(int i=0,j=4;i<4;i++,j++){cout<<"i="<<i<<" j="<<j;}

Ödev:

1. 0'dan 100'e kadar 5'er artan sayıların toplamını ve çarpımını bulan bir program yazınız.

(Algoritma veya akış şeması ile)

2. 1000'den 1'e kadar tersi ile düzü aynı olan sayıları bulan bir program yazınız.(For döngüsü kullanımı öğrenilmesi için özellikle 1000'den 1'e doğru verilmiştir.)

3. 89'dan 9'a 10 ar azalan sayılardan rakamları toplamı 5 ile bölünebilenleri ekrana yazdırın bir program yazınız.

Örnek: Dışardan girilen kişi sayısı ve kişilerin yaşlarını okuyarak yaş ortalamasını hesaplayan program yazınız.

[Döngü ile işlemler](#)

```
//This programme was created 11-11-11 as an example for the usage of FOR statement
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{int n;int yas;float ort=0;
cout<<"Kaç kişinin yaşı girilecek";
cin>>n;
for (int i=1;i<=n;i++)
{cout<<"\n"<<i<<" . kişinin yaşı giriniz\n";
cin>>yas;
ort+=yas;
}
cout<<n<<"kisinin yaşı ortalaması="<<(float) ort/n<<"\n";
system("pause");
return 0;
}
```

```
C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
Kaç kişinin yaşı girilecek5
1 . kişinin yaşı giriniz
23
2 . kişinin yaşı giriniz
34
3 . kişinin yaşı giriniz
45
4 . kişinin yaşı giriniz
67
5 . kişinin yaşı giriniz
6
5kisinin yaşı ortalaması=35
Devam etmek için bir tuşa basın . . .
```

Örnek: Dışarıdan girilen N adet notu okuduktan sonra aşağıdaki not değerlendirme sistemine göre sınıflayan bir program yazınız.

Harf not =

$not < 45, FF$ $45 \leq not < 55, FD$ $55 \leq not < 60, DD$ $60 \leq not < 70, DC$ $70 \leq not < 85, CB$ $not \geq 85, BA$

Not Hesabi

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int say,not,ba,cb,dc,dd,fd,ff;
int i;
int main()
{
    ba=cb=dc=dd=fd=ff=0;
    cout <<"Kaç not girilecek?\n";
    cin>>say;
    for (i=1;i<=say;i++){
```

```

cout<<i<<".notu giriniz";cin>>not;
if (not<45) ff+=1;
else if (not<55)
    fd+=1;
else if (not<60)
    dd+=1;
else if(not<70)
    dc+=1;
else if(not<85)
    cb+=1;
else ba+=1;
}
cout<<"\n";

cout<<ff<<"adet FF notu\n" <<fd<<"adet FD notu\n" <<
dd<<"adet dd notu\n" <<dc<<"adet dc notu\n" <<cb<< "adet cb notu \n"
"<<ba<<"adet ba notu girildi";
system("pause");
return 0;
}

```

```

C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
Ka  notu girilecek?
10
1.notu giriniz23
2.notu giriniz45
3.notu giriniz76
4.notu giriniz88
5.notu giriniz68
6.notu giriniz98
7.notu giriniz9
8.notu giriniz56
9.notu giriniz45
10.notu giriniz78

2adet FF notu
2adet FD notu
1adet dd notu
1adet dc notu
2adet cb notu
2adet ba notu girildiDevam etmek için bir tuşa basın . . .

```

7.5. while

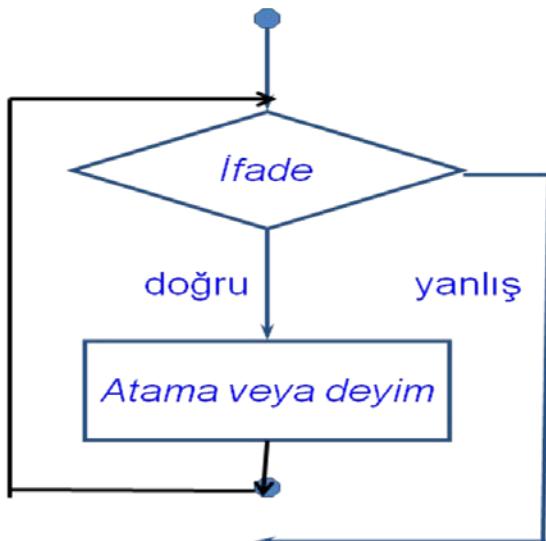
While (ifade) deyim; veya

While (ifade)

{deyim1;

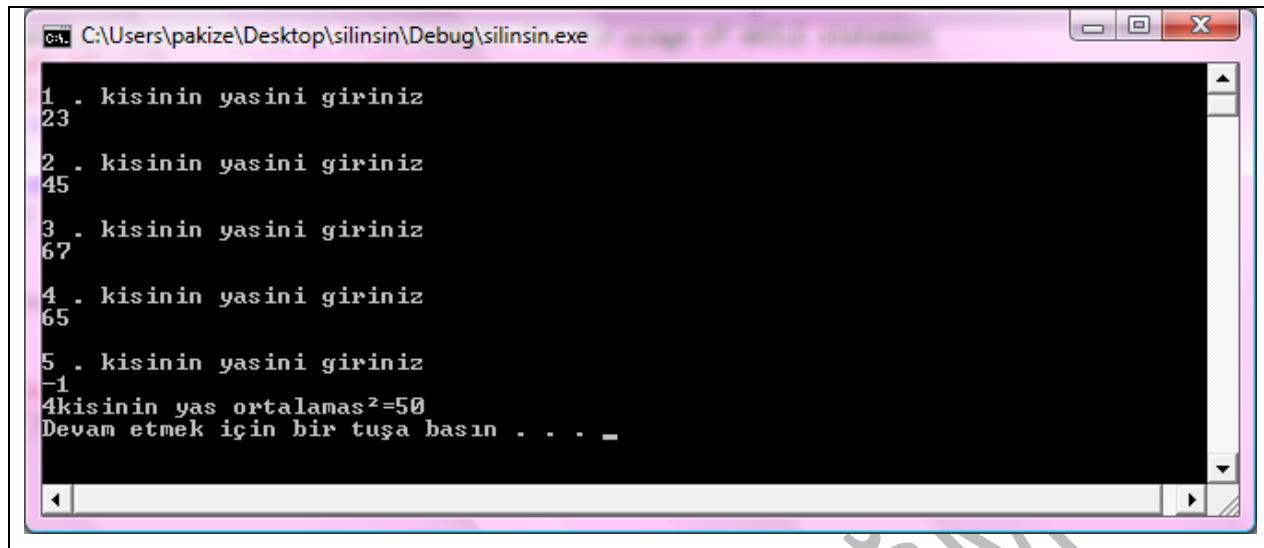
deyim2;

deyim3} şeklinde kullanılır. İfadedenin doğru olması durumunda while'den sonraki deyim veya deyim grupları işlenir. Burda tekrar sayısı for deyiminde olduğu gibi belli değildir. İfade doğru olduğu sürece deyim veya deyim grubu tekrar edecektir.



Örnek: Dışardan girilen kişilerin yaşlarını okuyarak yaş ortalamasını hesaplayan program. Kişi sayısı belli değil, eğer yaş değeri -1 girilirse okuma işlemi bitirilerek yaş ortalaması hesaplanıyor.

Kişi sayısının belli olmadığı durumda yaş ortalaması hesabı <pre> //This programme was created 11-11-11 as an example for the usage of WHILE statement #include "stdafx.h" #include <iostream> using namespace std; int _tmain(int argc, _TCHAR* argv[]) {int yas=0;float ort=0;int i=0; while (yas>=0) {ort+=yas; i=i+1; cout<<"\n"<<i<<" . kisinin yasini giriniz\n"; cin>>yas; } cout<<i-1<<"kisinin yas ortalamasi="<<(float) ort/(i-1)<<"\n"; system("pause"); return 0; }</pre>
--



```
C:\Users\pakize\Desktop\silinsin\Debug\silinsin.exe
1 . kisinin yasini giriniz
23
2 . kisinin yasini giriniz
45
3 . kisinin yasini giriniz
67
4 . kisinin yasini giriniz
65
5 . kisinin yasini giriniz
-1
4kisinin yas ortalaması=50
Devam etmek için bir tuşa basın . . . .
```

Ödev: Switch/case örneğinde yer alan dışardan girilen harfin sesli harf olup olmadığını bulan programı dışardan harf girildiği müddetçe n harf için gerçekleştiren programı while ifadesi ile yazınız.

7.6. Do/while

Do

{.....

.....}while (ifade)

İfade doğru olduğu sürece blok içindeki deyim gruplarını çalıştırın tekrarlı döngü yapısıdır. While'dan farklı döngü içine şartsız girilir. Döngü sonunda şart ifadesi yer alır.



Örnek: Dışardan girilen kişilerin yaşlarını okuyarak yaş ortalamasını hesaplayan program. Kişi sayısı belli değil, eğer yaş değeri -1 girilirse okuma işlemi bitirilerek yaş ortalaması hesaplanıyor.

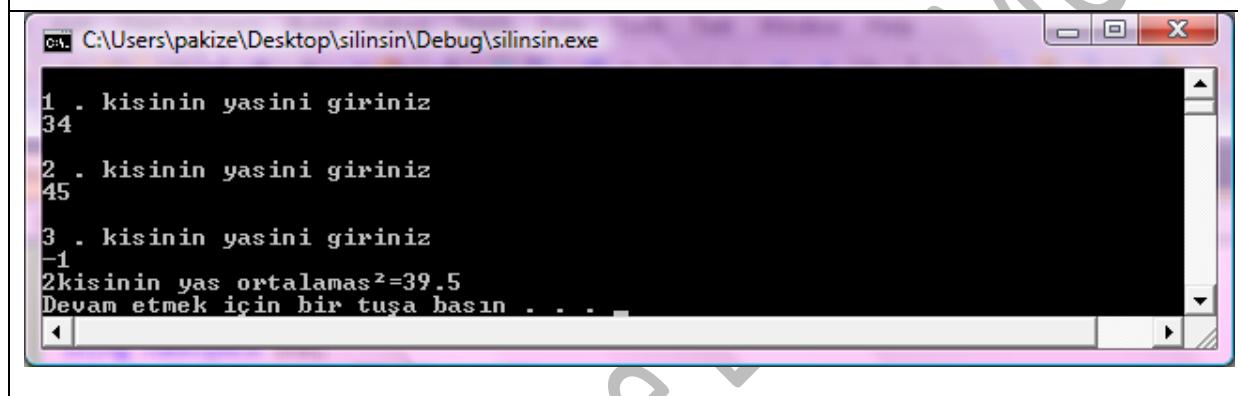
Do/while ile yaş ortalaması

//This programme was created 11-11-11 as an example for the usage of DO WHILE

```

statement
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{int yas=0;float ort=0;int i=0;
do
{ort+=yas;
i=i+1;
cout<<"\n"<<i<<" . kisinin yasini giriniz\n";
cin>>yas;
}while (yas>=0);
cout<<i-1<<"kisinin yas ortalaması="<<(float) ort/(i-1)<<"\n";
system("pause");
return 0;
}

```



7.7. break/continue

Bazı durumlarda döngü içinin atlanması istenir, bazen de döngüden çıkışması gerekebilir. Bu durumlar için break ve continue ifadeleri kullanılır.

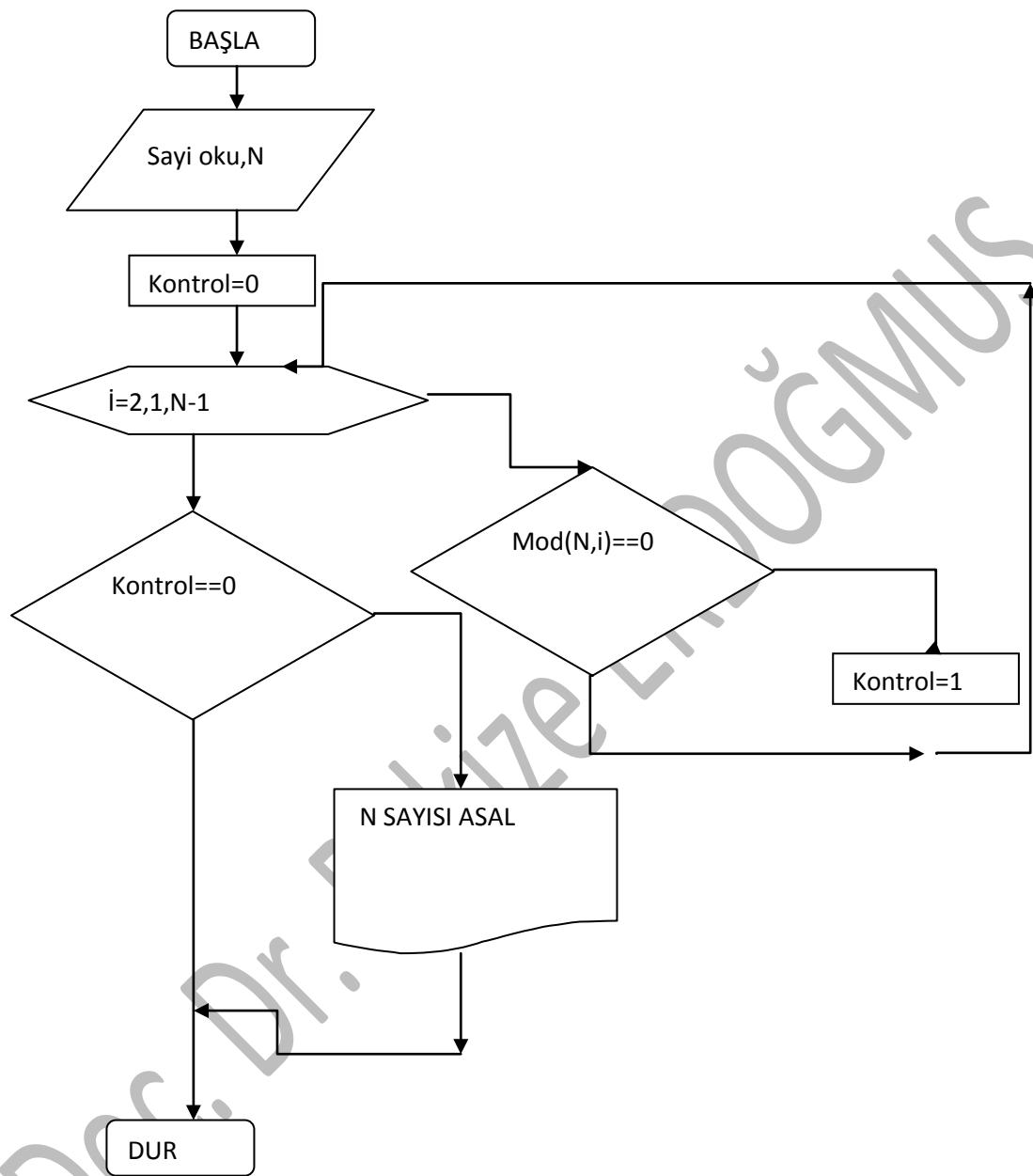
Break **while**, **for**, **do/while** döngü komutlarında ve **switch/case** şartlı dallanma komutunda kullanılır. Döngü içinden çıkmayı sağlar.

Continue ise yazıldığı yerden döngü sonuna kadar ki kodları işlemeden bir sonraki iterasyona geçer.

Rastgele Sayı Üretme Fonksiyonu

rand():0 ile RAND_MAX arasında rastgele bir sayı üretir.

Örnek: Bir ve kendinden başka böleni olmayan sayıya asal sayı denir. Bu tanımdan yola çıkarak dışardan girilen bir sayının asal olup olmadığını bulan bir program yazınız.



Break örneği

```

//This programme was created 11-11-11 as an example for the usage of BREAK statement
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{int N,kontrol=0;
cout<<"Asal olup olmadığını bulmak istediğiniz sayıyı giriniz\n";
cin>>N;
for (int i=2;i<N;i++)
{if (N-i*(float)(N/i)==0) {kontrol=1;break; }
if (kontrol==0) cout<<"\n"<<N<<" Sayisi Asal\n";
  
```

```

system("pause");
return 0;
}

c:\ C:\Documents and Settings\DuzceUnv\Desktop\sil2\Debug\sil2.exe
Asal olup olmadığını bulmak istediiniz sayınızı giriniz
599
599 Sayısı Asal
Devam etmek için bir tuşa basın . . .

```

Ödev: Programda küçük bir değişiklik yaparak dışarıdan sayı değeri -1 girene kadar girilecek sayıların asal olup olmadığını bulan bir program olmasını sağlayınız.

Ödev: 1'den dışarıdan girilen bir sayıya kadar ki sayıların asal olup olmadığını bulan bir program yazınız.

Ödev: Aşağıdaki program dışardan girilen bir sayı 5'e bölünüyor ise mesaj veren bir program olarak düşünülmüştür. **Ancak bir mantık hatası vardır.** Programı inceleyerek neden hata verdiğiini bulunuz.

```

//This programme was created 11-11-11 as an example for the usage of CONTINUE
statement
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int N;
cout<<"Bir tam sayı giriniz?";
cin>>N;
while (N>0)
{if (N/5.0!=int(N/5.0)) continue;
cout<<"Sayı 5'e bölünüyor";
cout<<"\nBir tam sayı giriniz?\n";
cin>>N;}
system("pause");
return 0;
}

```

UYGULAMA 1: Program çalıştırıldığında bilgisayar rastgele bir sayı üreticek. Kullanıcı ise dışardan değerleri girerek bilgisayarın tuttuğu sayıyı tahmin edecek. Bilgisayar kullanıcının sayısı ile tuttuğu sayiyi karşılaştırıp küçük veya büyük diyerek kullanıcının sayiyi bulmasına yardım edecek, kullanıcı bilgisayarın tuttuğu sayiyi bilince program son bulacak.

ÖN BİLGİ:

rand() fonksiyonu ve srand() fonksiyonu

rand() fonksiyonu 0 ile RAND_MAX arasında rastgele sayı üretiminde kullanılır. Üretilen maximum sayı RAND_MAX sabitinde tutulur.

Rastgele sayı üretimi bir çekirdek sayı(seed) ile başlar ve diğer sayılar bu çekirdek sayının çeşitli değerlere göre mod alması sonucu oluşturulur. Dolayısıyla eğer farklı bir çekirdek verilmez ise program her seferinde aynı ilk değeri baz alarak program içinde rastgele sayı üretimine başlar. Program içerisinde n kez üretilen rastgele sayılar birbirinden farklı olur. Fakat program her run edilişinde aynı değerleri üretir. Bu durumdan kurtulmak için srand() kullanılır.

1. Belli aralıkta sayı üretimi(alt,ust)

$x=alt+rand()%(ust-alt+1)$ ile elde edilir.

5-15 arası sayı üretimi için;

$x=5+rand()%(10+1);$ kullanılır.

Eğer rand() 0 sayısı üretirse: $x=====5+0=5$

Eğer rand() 1 sayısı üretirse: $x=====5+1=6$

.....

Eğer rand() 21 sayısı üretirse: $x=====5+10=15$

Eğer rand 22 sayısı üretirse: $x=====5+0=5$ üretir. Yani tekrar başa döner.

2. Eğer 0-1 arası sayı üretilmek istenirse;

(double) rand()/RAND_MAX kullanılır.

Eğer rand() 0 üretirse; $0/RAND_MAX=0$

Eğer rand() RAND_MAX üretirse; RAND_MAX/ RAND_MAX=1 üretir.

strnd(): Her seferinde farklı sayı üretiminde kullanılır. Time fonksiyonu ile birlikte kullanılırsa sistem saatine göre her seferinde değişen sayılar elde edilir.

strnd((unsigned int) time(0)) şeklinde kullanılır.

#include <time.h> kütüphanesi eklenmelidir.

ÖRNEK:

Rastgele Sayı Üretilimi

```
//Rastgele sayı üretimi
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int s;
    cout<<"1. rastgele sayı= "<< rand()<<endl;
    cout<<"2. rastgele sayı"<< rand()<<endl;
    system("pause");return 0;
}
```

Bu programı kaç kere çalıştırırsanız çalıştırın aynı sayıları üretir.

ÖRNEK: Her çalışmada değişen sayılar üreten program

```
// Her seferinde farklı rastgele sayı üretimi
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int s;
    srand((unsigned int) time(0));
    cout<<"1. rastgele sayı= "<< rand()<<endl;
    cout<<"2. rastgele sayı"<< rand()<<endl;
    cout<<"1-100 arası"<<rand()%100<<endl;
    cout<<"20-80 arası"<<20+rand()%60<<endl;
    cout<<"0-1 arası"<< (double) rand()/RAND_MAX;
    system("pause");return 0;
}
```

UYGULAMA 1:

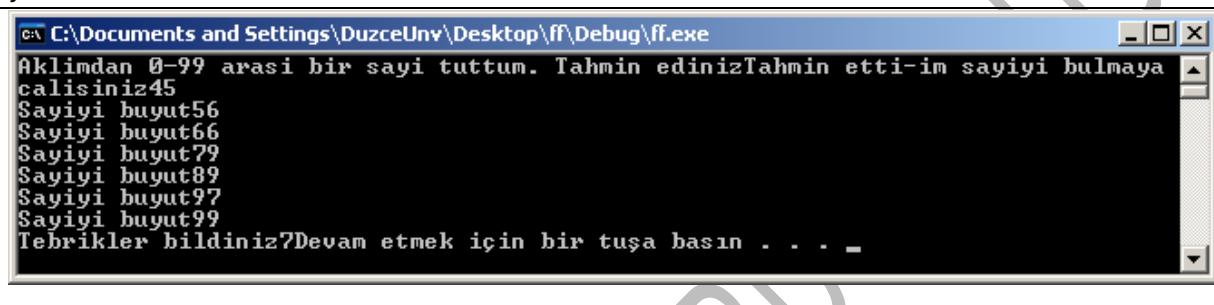
Uygulama 1:Sayı tahmin

```
/* Sayı bulma Oyunu*/
#include "stdafx.h"
#include <iostream>
#include <ctime> //Zaman fonk içeren C kütüphanesi
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    int sayı,tah=100,tek;
    srand((unsigned int) time(0)); //Her seferinde farklı sayı üretmek için kullanılır.
```

```

sayi=rand()%100;//rand() Rastgele sayı üretmeye yarayan bir fonksiyon
cout<<"Aklimdan 0-99 arası bir sayı tuttum. Tahmin ediniz";
cout<<"Tahmin ettiğim sayıyi bulmaya çalışınız";
tek=0;
while (tah!=sayi)
{tek=tek+1;
cin>>tah;
if (tah<sayi) cout<<"Sayiyi buyut";
if (tah>sayi) cout<<"sayiyi kucult";
}
cout<<"Tebrikler bildiniz"<<tek<<" kerede bildiniz";
system("pause");
}

```



UYGULAMA 2: Bilgisayar tarafından sırası ile bir basamaklı sayı üretiliyor ve bilgisayar sayıyı birkaç saniye gösterdikten sonra ekranдан siliyor. Ve size sayıyı girmenizi istiyor. Eğer sayıyı doğru girerseniz, iki basamaklı gösteriyor. Böylece hatırladığınız ve doğru girdiğiniz sürece basamak sayısını artırıyor. Ve hata yaptığından kaç haneye kadar geldi iseniz size bir IQ seviyesi gösteriyor.

UYGULAMA 2:IQ_test

```

// IQ_test.cpp : This programme was designed by Pakize ERDOGMUS for an example of a
simple game//20_04_2011
#include "stdafx.h"
#include <iostream>
#include <cctime>
using namespace std;
int main() {
    cout<<"\t\tBu program bir hafiza test oyunudur.\n\n\n\t Acaba kısa süreli olarak en
fazla kaç haneye\n\n \tkadar sayıyi aklinizda tutabilirsiniz?\n ";
    int sayi_blg,sayi_user,sayi_mod=1,j;
    int i,k,puan=0,sayı;
    j=10;
    cin>>sayı;
    srand((unsigned int) time(0));
    for ( i = 1; i <=sayi; ++i)
    {   sayi_mod=sayı_mod*10;
        sayı_blg=rand()*1234934;
        sayı_blg=abs(sayı_blg%sayı_mod);
        cout<<"Sayi= "<<sayı_blg<<endl;
        for (k=1;k<800000000;++k){}
        system("cls");
        cout<<"\nBiraz önceki sayıyi hatırlayınız\n";
    }
}

```

```

    cin>>sayi_user;
    if (sayi_user==sayi_blg) puan=puan+1; else break;
    switch (puan)
    {case 4:
    cout<<"\n\tDusuk IQ";
    break;
    case 5:
    cout<<"\n\tNormal altı";
    break;
    case 6:
    cout<<"\n\tNormal";
    break;
    case 7:
    cout<<"\n\tZeki";
    break;
    case 8:
    cout<<"\n\tSuper";
    break;
    case 9:
    cout<<"\n\tDahi";
    break;
    default: cout<<"\n\tYa dahi otesi veya çok dusuk IQ'ya sahipsiniz.\n\t Ama her iki
    durumda da anormalsiniz.\n\n";
    }
    system("pause");
    return 0; }
```



UYGULAMA 3: 100 ile 999 arası sayılardan Armstrong sayıları bulunuz.

Not: Rakamlarının küpleri toplamı kendisini veren sayıya Armstrong sayı denir.

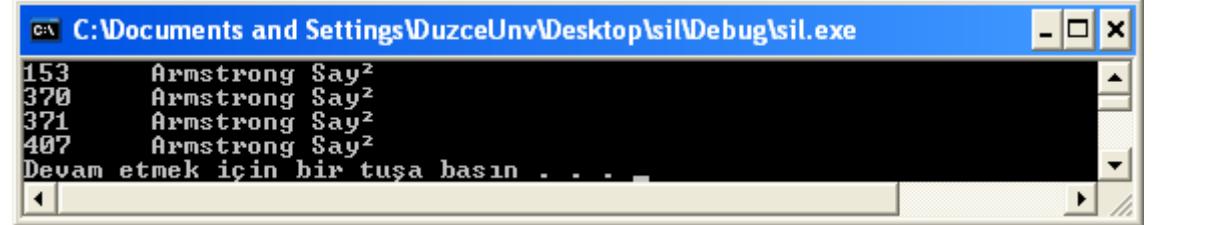
$$1^3 + 5^3 + 3^3 = 153$$

```

armstrong_sayı
#include <iostream>
#include <string>
using namespace std;
void main()

{int a;
//100-999 arası sayılar
int yuzler,onlar,birler,i;
for (a=100;a<=999;a++)
    {yuzler=(a-a%100)/100;
    onlar=(a%100-a%10)/10;
    birler=a-yuzler*100-onlar*10;
    if (a==birler*birler*birler+onlar*onlar*yuzler*yuzler) cout<<a<<"Armstrong Sayı"<<endl;}
```

```
        system("Pause");
}


```

Doc. Dr. Pakize ERDOĞMUŞ

BÖLÜM SONU SORULAR VE UYGULAMALAR

Aşağıdaki program kesimlerinin çalışması sonucu değişken değerleri ne olur?

1. int a=6,b=56
if (a<b) a=a+b;
b=b+6;
 2. int a=23,b=45,c=17;
if (a!=(b-c))
{a=a+2;b+=a+c;}
a+=b-c;
 3. int a=13,b=14,c=37;
if (a<=(b-c))
{a+=2;b+=a-c;}
else if(a==(b-c)) { a+=b-c; b+=3;}
else {b=b*a;}
b=a+b+c;
a+=c;
b+=6;
 4. // This programme was developed for the do/while application on November the 14th 2011.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char ch;
char cevap;
do {
    cout << "Karariniz (e, h): ";
    if (cin >>cevap)
        cevap = tolower(cevap); //tolower(degisken) degisken karakterini kucuk harfe cevirir.
    else
        cevap = 'h';
} while ((cevap != 'e') && (cevap != 'h'));
system("pause");
return 0;
}
```
- Yukarıdaki programın ne iş yaptığıni bulunuz.

8. C++'ta Diziler

Diziler bilgisayar programlamanın önemli konularındandır. Dışarıdan okutulacak bir veya iki elemanımız varsa bunları cin ile okutmak kolaydır. Ancak dışarıdan girilecek 100 değer var ise ne yapacağız? Bunun en kolay yöntemi dizi tanımlamaktır.

Aynı türde birden fazla verinin tek isim altında saklanmasını sağlayan yapıya **dizi** denir. Belli sayıda verinin bellekte saklandığı değişken listeleridir. Bu liste tek bir isimle program içinde kullanılabilir.

Dışarıdan girilecek veri sayısı arttıkça her bir değişkene farklı bir isim vermek değişken sayısını dolayısıyla programın karmaşıklığını artıracaktır. Diziler yardımıyla çok sayıda değişkeni aynı isimle sadece farklı indislerle saklamak mümkündür.

a dizisi oluşturmak demek;

`a[0], a[1], a[2], ..., a[n]` gibi farklı indislere sahip bir değişkenler topluluğu oluşturmak demektir. Bu değişkenlerin tipi aynıdır. C++'ta ve C'de diziler köşeli parantez ile, Visual Basic ve Matlab'da parantez ile gösterilirler. Diziler çok sayıda değişken kullanılması gereken ve Matematiksel dizi işlemleri gereken durumlarda kullanılır. Önceki konularda dışarıdan girilen `n` sayının ortalamasını bulan bir program yazmıştık. Acaba bu `n` sayıyı büyükten küçüğe sıralamak gerekirse `n` adet değişken ismi ile bu işi gerçekleştirmek zordur. Daha önce üç sayı bile bu işlemin oldukça zahmetli olduğu görülmüştü.

`int x, y, z, t, m, n, p, r, s` yerine;

`int A[8]` ifadesi sonucu;

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
------	------	------	------	------	------	------	------

şeklinde bilgisayarda yer açılacaktır.

Genel tanım:

Tip dizi_adi[boyutu] şeklindedir.

Örneğin ;

`int a[5]; char p[7]; float sayı[6];`

şeklinde dizi tanımı yapılır.

`int a[5]` tanımı ile `a[0]`'dan `a[4]`'e kadar 5 adet tamsayı değişken saklayacağımız yer tanımlamış oluruz. Bu 5 değişkene aynı isim ile sadece farklı indisler ile ulaşırız.

7.1. Dizilere ilk değer Atamak

Program içerisinde dışarıdan değilde program tarafından diziye değer atanabilir. Bunu iki şekilde yapabiliriz. Tek tek değer atama veya topluca değer atama.

Örneğin `int a[5]` şeklinde tanımladığımız diziye çeşitli şekillerde atama yapabiliriz.

- `a[0]=2; a[1]=4;a[3]=65;` gibi değer atayabiliriz. Veya
- `a[]={2,4,65}` gibi değer atayabiliriz.

Bu durumda dizi boyutu `[]` içerisinde verilmemişinden kıvırcık parantez içerisinde virgül ile ayrılmış her bir değişken sayısı kadar eleman tutan bir dizi olacaktır. Yani 3 elemanlı bir dizi oluşturur. Dizinin elemanları ve değerleri

`a[0]=2;a[1]=4;a[2]=65;` olur.

- `a[3]={2,64}` gibi değer atayabiliriz.

Bu durumda `a[0]=2, a[1]=64` olur. C++ tarafından `a[2]=0` atanır.

Not: Dizi boyutu ya `4,5,9,...` gibi sabit bir tamsayıdır veya tanımlanmış bir sabit değerdir.

NOT:

- `int a[5];` veya
- `const int n=5;
int a[n];`
şeklinde tanımlanır.
- `int n=5;
int a[n];` şeklinde
tanımlanamaz.

Bir dizi tanımlandığında dizi elemanları hafizada yanyana bellek gözlerine yerlesirler. Dizinin ismi ise bir dizinin hafızadaki ilk değerinin adresini gösterir. Yani `int a[4];` dizisi tanımlandığında `a a[0]`'ın adresini gösterir.

<code>a[0]</code>						
<code>a[1]</code>						
<code>a[2]</code>						
<code>a[3]</code>						

--	--	--	--	--	--	--

char dizilerde dikkat edilmesi gereken nokta satır sonu karakteridir. Örneğin;

char a[6] = "Pakize" ataması ["IntelliSense: a value of type "const char \[7\]" cannot be used to initialize an entity of type "char \[6\]" c:\users\pakize\desktop\mmm\mmm\mmm.cpp"](#) hatasına yol açar. Doğru atama char a[7] = "pakize" dir. Dizi elemanlarının yerleşimi aşağıdaki gibidir.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
'p'	'a'	'k'	'i'	'z'	'e'	'\0'

Örnek: Aşağıdaki dizi tanımlarını inceleyerek ekran çıktısını tahmin ediniz.

```
// degisken atama ile
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{const int n=5;
 double x[n];
 int y[3];
 float a[]={2,4,5,8,9};
 char b[20] = "Pakize";
 x[1]=4;x[0]=34.40;
 y[2]=23;
 cout<<"x dizisinin eleman sayisi n=" <<n<<endl;
 cout<<"x[0] = "<<x[0]<<endl;
 cout<<"x[1] = "<<x[1]<<endl;
 cout<<"y dizisi eleman sayisi 3" <<endl;
 cout<<"y[2] = "<<y[2]<<endl;
 cout<<"a dizisi eleman sayisi 5" <<endl;
 cout<<"a[0] = "<<a[0]<<endl;
 cout<<"a[1] = "<<a[1]<<endl;
 cout<<"b dizisi eleman sayisi 20" <<endl;
 cout<<"b[0] = "<<b[0]<<endl;
 cout<<"b[1] = "<<b[1]<<endl;
 cout<<"b[2] = "<<b[2]<<endl;
 cout<<"b[3] = "<<b[3]<<endl;
 cout<<"b[4] = "<<b[4]<<endl;
 cout<<"b[15] = "<<b[15]<<endl;
system("pause");
}
```

```
c:\Documents and Settings\...\\Desktop\ttt\Debug\ttt.exe
x dizisinin eleman sayısı n=5
x[0]=34.4
x[1]=4
y dizisi eleman sayısı 3
y[2]=23
a dizisi eleman sayısı 5
a[0]=2
a[1]=4
b dizisi eleman sayısı 20
b[0]=P
b[1]=a
b[2]=k
b[3]=i
b[4]=z
b[15]=
Devam etmek için bir tuşa basın . . .
```

Örnek: Dizi girişi ve ekran çıktısı 5 adet sayıyı dizi tanımı ile okutup ekrana yazdırınız.

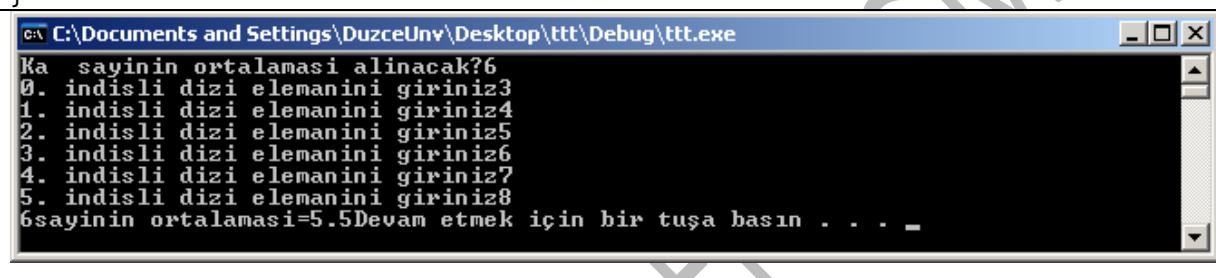
```
//temel_dizi.cpp
//Bu program dizi giriş çıkışını yapar.
//03.03.2011
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{int a[5];
int i,top=0;
cout<<"Dizi elemanları girişi"<<endl;
for (i=0;i<5;i++)cout<<"Dizinin a["<<i<<"]. elemanın değerini giriniz="; cin>>a[i];
for (i=0;i<5;i++){cout<<"Dizinin a["<<i<<"]. elemanın değeri="<<a[i]<<endl;
top=top+a[i]; }
system("pause");
cout<<"      "<<endl;
cout<<"Elemanlar toplamı top= "<<top; system("pause");}
```

```
c:\Documents and Settings\...\\Desktop\silin\Debug\silin.exe
Dizi elemanları girişi
Dizinin a[0]. elemanın değerini giriniz=2
Dizinin a[1]. elemanın değerini giriniz=3
Dizinin a[2]. elemanın değerini giriniz=4
Dizinin a[3]. elemanın değerini giriniz=5
Dizinin a[4]. elemanın değerini giriniz=6
Dizinin a[0]. elemanın değer=2
Dizinin a[1]. elemanın değer=3
Dizinin a[2]. elemanın değer=4
Dizinin a[3]. elemanın değer=5
Dizinin a[4]. elemanın değer=6
Elemanlar toplamı top= 20 Devam etmek için bir tuşa basın . . .
```

Örnek: Dışardan girilen n sayının ortalamasını dizi kullanarak hesaplayan bir program yazalım.

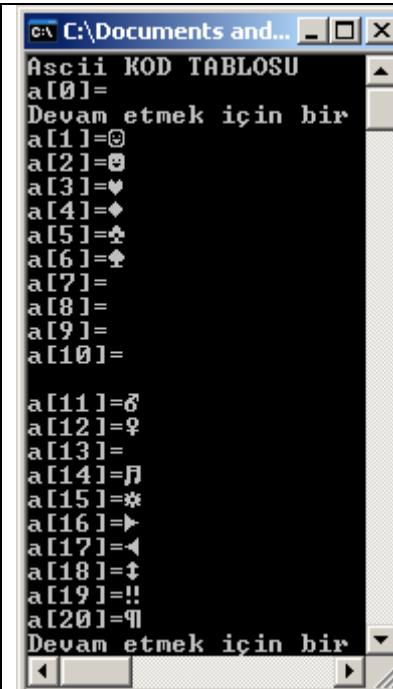
```
//This programme was created 11-11-11 as an example for the usage of ARRAYS
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{int N;int a[20];float toplam=0;
cout<<"Kaç sayinin ortalamasi alınacak?";
cin>>N;
for (int i=0;i<N;i++)
    {cout<<i<<". indisli dizi elemanini giriniz";
cin>>a[i];
toplam+=a[i];}
cout<<N<<"sayinin ortalamasi="<

```



Örnek: 255'e kadar ASCII karakterlerini dizi ile gösteren bir program yazınız.

```
// diziler.cpp : main project file.
//Bu program ASCII karakterlerini gösterir.
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{char a[255];
int i;
printf("Ascii KOD TABLOSU\n");
for (i=0;i<255;i++)
    {a[i]=i;
cout<<"a["<<i<<"]="<<a[i]<<endl;
if (i%20==0) system("pause");}
system("pause");
}
```



Örnek: Değerlerini girdiğiniz bir dizinin en küçük ve en büyük elemanlarını bulunuz.

```
// ilk değerli diziler.
//03.03.2011
```

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{int a[]={23,34,54,65,21,98,87,67,77,65,43,5,67};
int i,ek=a[0],eb=a[0];
for (i=0;i<12;i++)
    {if (a[i]<ek) ek=a[i];
if (a[i]>eb) eb=a[i];}
cout<<"En küçük sayı= "<<ek<<endl;
cout<<"En büyük sayı= "<<eb<<endl;
system("pause");}

```

```
c:\Documents and Settings\DuzeUnv\Desktop\cpp\ilk_degerli_diz\Debug\ilk_degerli_diz.exe
En küçük sayı= 5
En büyük sayı= 98
Devam etmek için bir tuşa basın . . . .
```

Örnek: Bir zarın 120 kez atılması durumunda kaç kez 1,2,..6 geleceğini simule eden program.

```
//Bu program bir zarın 100 kere atıldığından kaç defa hangi sayı geleceğini simule
eder.
//03.03.2011
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
void main()
{int d=0;int a[]={0,0,0,0,0,0,0};
int i;
srand(time(NULL));
for (i=1;i<=120;i++)
    { //1-6 arası bir sayı üretir
        d=rand()%6+1;
        cout<<d;
        a[d]++;
    }
//Üretilen sayıları göster
for (i=1;i<=6;i++) cout<<i<<"den "<<a[i]<<" defa üretildi"<<endl;
system("pause");}

```

```
c:\Documents and Settings\DuzeUnv\Desktop\cpp\zar_dizi\Debug\zar_dizi.exe
62662565412634256566156121455441325243212336232645644544313223144336265324426652
2266113111155211623166164266621424532231den 20 defa 3retildi
2den 26 defa 3retildi
3den 16 defa 3retildi
4den 18 defa 3retildi
5den 15 defa 3retildi
6den 25 defa 3retildi
Devam etmek için bir tuşa basın . . . .
```

Örnek: Bir T serisi elemanları $0, 1, 2, 3, 11, 31, \dots$ şeklindedir. Yani $n > 4$ için kuralı aşağıda verilmiştir.

$$T_n = \begin{cases} n & n = 1 \\ n & n = 2 \\ n & n = 3 \\ n & n = 4 \\ n = 2(T_{n-1} + T_{n-2}) + T_{n-3} + T_{n-4} & n = 5 \end{cases}$$

Terim sayısı $n=30$ için bu serinin elemanlarını hesaplayıp ekrana yazdırın bir program yazınız.

```
// diziler ile çözüm
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{int a[30]={0,1,2,3};
for (int i=4;i<30;i++)
{a[i]=2*(a[i-1]+a[i-2])+a[i-3]+a[i-4];
cout<<"a["<<i<<"]="<<a[i]<<endl;
system("pause");
}

cv C:\Documents and Settings\DUZCEUNIV\Desktop\ttt\Debug\ttt.exe
a[4]=11
a[5]=31
a[6]=89
a[7]=254
a[8]=728
a[9]=2084
a[10]=5967
```



```
// degisken atama ile
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{int a=0,b=1,c=2,d=3,e;
for (int i=5;i<=30;i++)
{e=2*(c+d)+a+b;
cout<<"Dizinin "<<i<<". elemani="<<e<<endl;
a=b;b=c;c=d;d=e;}
system("pause");
}

cv C:\Documents and Settings\DUZCEUNIV\Desktop\ttt\Debug\ttt.exe
Dizinin 5. elemani=11
Dizinin 6. elemani=31
Dizinin 7. elemani=89
Dizinin 8. elemani=254
Dizinin 9. elemani=728
Dizinin 10. elemani=2084
Dizinin 11. elemani=5967
Dizinin 12. elemani=17084
```

BÖLÜM SONU SORULAR VE UYGULAMALAR

1. Bir futbol takımının oynadığı maçları tutan bir dizi oluşturunuz. Bu takımın kazandığı maç için(3), kaybettiği için(0) ve berabere kaldığı maç için(1) puan alacaktır. Örnek bir dizi aşağıda verilmiştir.

$FB[0]=0$ $FB[1]=1$ $FB[2]=3$ $FB[3]=3$ $FB[4]=3$ $FB[5]=3$ $FB[6]=1$ $FB[7]=0$

Bu şekilde önce kaç maç oynadığını dışarıdan alan(MS), daha sonra maç sayısı kadar sonucu kullanıcıdan isteyen ve bu sonuçlara göre yukarıdaki gibi bir dizi oluşturan, sonuç olarak takımın kaç puanı olduğunu yazdırın bir program yapınız. Örnek çıktı aşağıdaki gibidir.

```
C:\Documents and Settings\UzuceUnv\Desktop\sil2\Debug\sil2.exe
Bu program bir takimin toplam puanini hesaplar
Takimin ismini giriniz :Galatasaray
Bu takimin oynadigi mac sayisini giriniz5
0. mac sonucunu giriniz:<a-Galibiyet,b-Maglubiyet,c-Beraberlik>
a
1. mac sonucunu giriniz:<a-Galibiyet,b-Maglubiyet,c-Beraberlik>
a
2. mac sonucunu giriniz:<a-Galibiyet,b-Maglubiyet,c-Beraberlik>
c
3. mac sonucunu giriniz:<a-Galibiyet,b-Maglubiyet,c-Beraberlik>
a
4. mac sonucunu giriniz:<a-Galibiyet,b-Maglubiyet,c-Beraberlik>
a_

```

```
C:\Documents and Settings\UzuceUnv\Desktop\sil2\Debug\sil2.exe
0. mac sonucu :3
1. mac sonucu :3
2. mac sonucu :1
3. mac sonucu :3
4. mac sonucu :3
Galatasaray takiminini toplam puanı=13 dir.Devam etmek için bir tuşa
` - =
```

2. Değerleri 1-100 arası olacak şekilde üretilen bir rastgele sayı dizisinin içinde değeri 50'den büyük ve eşit kaç sayı olduğunu hesaplayan bir program yazınız?

9. İki Boyutlu Diziler

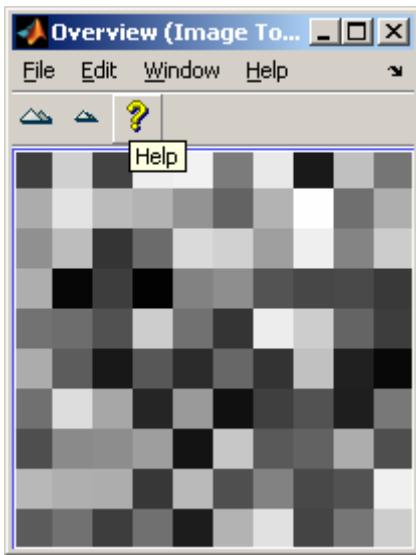
İki boyutlu dizilerin en önemli kullanım yeri matrislerdir. Matrislerde toplama, çarpma işlemlerini iki boyutlu diziler ile gerçekleştirilir. Bunun dışında da birçok pratik uygulama da çok boyutlu dizi tanımı gereklidir.

Diziler birçok pratik uygulamanın temelini içerirler. Örneğin ses bilgisayar için tek boyutlu bir dizi, görüntü ise iki boyutlu bir dizidir. Eğer ses,müzik, görüntü gibi veriler sayısal hale getirilirse((Örnekleme+Kuantalama)(Sampling+Quantization)) işlenebilir. Dolayısıyla sayısal bir görüntüyü işlemek iki boyutlu matris işlemi, sesi işlemek ise bir boyutlu matris işlemidir. Bir sesin yükseltilmesi dizi değerlerinin arttırılması, hızlandırılması bir saniye içinde hoparlöre gönderilecek eleman sayısının artırılması demektir.

$X=[0.8147 \quad 0.9058 \quad 0.1270 \quad 0.9134 \quad 0.6324 \quad 0.0975 \quad 0.2785 \quad 0.5469 \quad 0.9575]$ şeklindeki bir dizi Matlab için birçok anlama gelir. X bir sayısal ses örneği olabilir. Bu değerleri sound ile hoperlore gönderdiğimizde ses olarak işitiriz. Burda önemli olan frekanstır. Yani hoperlor bir saniyede bu değerlerden kaç tanesini ses olarak verecektir.

144	29	16	180	217	85	192	7	1	238
28	170	208	151	94	18	232	185	112	101
66	17	149	240	142	12	48	71	180	39
84	216	79	40	196	114	27	241	57	179
79	41	30	189	115	20	161	159	230	153
63	194	182	26	38	197	244	121	4	54
33	64	77	206	70	63	177	163	49	88
9	86	89	154	39	46	226	228	181	228
158	223	59	163	206	137	21	86	195	164
189	193	178	26	140	226	215	4	52	251

Yukarıda verilen 10X10 boyutlarındaki Y matrisi ise bir görüntü matrisidir. Bu matrisi ekran kartına gönderdiğimizde görülecek resim aşağıda verilmiştir. Bu resim ve matristen anlaşılıcığı üzere birinci satırda 192 sayısı ekran kartında beyaza yakın bir renge, 7 sayısı da siyaha yakın bir renge dönüştürülmüür. (0-Siyah,255-Beyaz)



C++ için çok sayıda görüntü işleme kütüphanesi mevcuttur. Aşağıdaki web sayfasından bu kütüphanelerden biri indirilebilir. <http://cimg.sourceforge.net/> The Cimg kütüphanesi, bir görüntüyü alıp, kaydetmek, işlemek ve saklamak için gerekli fonksiyon ve sınıfları içerir.

İkiboyutlu bir dizinin genel tanımı:

Tip dizi_adı[boyut1][boyut2] şeklindedir.

Örneğin int a[3][3] şeklinde dizi tanımı yapıldığında a[0][0]'dan a[2][2]'ye kadar 9 adet tamsayı değişken saklanabilecek bellek bölgesi tanımlanmış olur. Bu 9 değişkene aynı isim ile sadece farklı indisler ile ulaşılır.

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]

8.1. İki Boyutlu Sayısal Dizilere İlk Değer Atama

Tek boyutlu dizilere değer atamadan farklı olarak, iki boyutlu dizilere ilk değer atanırken sütun sayısını gösteren köşeli parantez içeriği boş bırakılamaz.

```
int A[4][3]={ {1,2,3},{3,4,5},{5,6,7},{7,8,9}};
```

```
int A[][3]={ {1,2,3},{3,4,5},{5,6,7},{7,8,9}};
```

```
int A[4][3]={1,2,3,3,4,5,5,6,7,7,8,9};
```

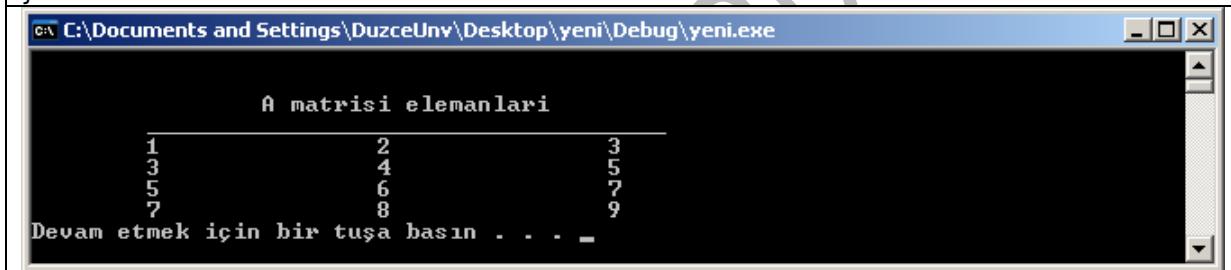
veya int A[4][3]; A[0][0]=1;A[0][1]=2;.... şeklinde atama yapılan dizi 4 satır, 3 sütunlu bir matris olacaktır. Bu matrisin matematiksel gösterimi aşağıda verilmiştir.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \\ 7 & 8 & 9 \end{bmatrix}$$

Örnek: Yukarıdaki matrisin elemanlarını ekranaya yazdırın bir program.

iki boyutlu dizi örneği

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{
    int a[4][3]={{1,2,3},{3,4,5},{5,6,7},{7,8,9}};
    cout<<"\n\n\tA matrisi elemanlari\n";
    cout<<"\t_____ \n";
    for (int i=0;i<4;i++)
    {
        for (int j=0;j<3;j++)
            cout<<"\t"<<a[i][j]<<"\t";
        cout<<"\n"; //Matris formatinda yazdirmak icin
    }
    system("pause");
}
```



Örnek: N boyutlu kare iki matrisin toplamını yapan program.

N boyutlu iki matrisin toplamını alan program

```
// matris toplamı
#include "stdafx.h"
#include <iostream>
using namespace std;
void main()
{int a[10][10],b[10][10],c[10][10],N,i,j;
 cout<<"Bu program max 10 boyutlu iki matrisin toplamını alır"\  
       << endl;
 do
 {
 cout<<"Matris boyutunu giriniz N max 10 olabilir";
 cin>>N;
 }while(N<0||N>10);
 for(i=0;i<N;i++)
 {
     for(j=0;j<N;j++)
     {cout<<"birinci ve ikinci matris elemanlarını giriniz"\  
           << endl;
      cout<<"a["<<i<<"]["<<j<<"]. eleman ile"\  
           << " b["<<i<<"]["<<j<<"].
eleman değerleri"\  
           << endl;
      cin>>a[i][j]>>b[i][j];
      cout<<endl;
 }
 //Toplama işlemi
 for(i=0;i<N;i++)
 {
     for(j=0;j<N;j++)
     {a[i][j]=a[i][j]+b[i][j];
      cout<<"Toplam"\  
           << i << " " << j << " : " << a[i][j] << endl;
 }
 }
```

```

    {c[i][j]=a[i][j]+b[i][j];cout<<c[i][j]<<" ";}
    cout<<endl;
}
system("pause");
}

c:\Documents and Settings\DuzceUnv\Desktop\dizi\Debug\dizi.exe
Bu program max 10 boyutlu iki matrisin toplam2n2 alır
Matris boyutunu giriniz N max 10 olabilir2
birinci ve ikinci matris elemanlarını giriniz
a[0][0]. eleman ile b[0][0]. eleman değerleri
1 2

birinci ve ikinci matris elemanlarını giriniz
a[0][1]. eleman ile b[0][1]. eleman değerleri
3 4

birinci ve ikinci matris elemanlarını giriniz
a[1][0]. eleman ile b[1][0]. eleman değerleri
5 6

birinci ve ikinci matris elemanlarını giriniz
a[1][1]. eleman ile b[1][1]. eleman değerleri
7 8

3 7
11 15
Devam etmek için bir tuşa basın . . .

```

Örnek: NXN boyutlu bir kare matrisin asıl köşegeni üzerindeki elemanların toplamını alan program.

```

Esas kosegenlerin toplamini yapan program
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
void main()
{
int a[10][10],b[10][10],c[10][10], i,j,N,toplam=0;
cout<<"Bu program değerleri 1-5 arası olan bir rastgele matrisin esas köşegeni
üzerindeki elemanlar toplamını bulur"<<endl;
do
{cout<<" NXN lik matrisin N boyutunu giriniz"<<endl;
cin>>N;
}while(N<0||N>10);
//Elemanların üretimi ve esas köşegen toplamı
for(i=0;i<N;i++)
    for (j=0;j<N;j++)
        {a[i][j]=rand()%5;if(i==j) toplam+=a[i][j];}
//Yazdırma
for(i=0;i<N;i++)
    {for (j=0;j<N;j++)
        cout<<a[i][j]<<" ";cout<<endl;}
cout<<"Esas kosegen üzerindeki elemanlar toplamı="<<toplam<<endl;
system("pause");
}

```

```
C:\Users\pakize\Desktop\pppp\Debug\pppp.exe
Bu program de-erleri 1-5 aras2 olan bir rastgele matrisin esas kosegen elemanlar toplamini bulur
N*M lik matrisin N boyutunu giriniz
5
1 2 4 0 4
4 3 3 2 4
0 0 1 2 1
1 0 2 2 1
1 4 2 3 2
Esas kosegen üzerindeki elemanlar toplami=9
Devam etmek için bir tuşa basın . . .

```

Soru: Ters köşegen üzerindeki elemanları toplatmak için kullanılacak if ifadesi ne olurdu?

Cevap: Ters köşegen üzerindeki elemanları toplatmak için;
if(j==(N-1-i))) ifadesi kullanılır.

Örnek: MXN ve NXN boyutlarındaki iki matrisin çarpımını alan program.

Matris Çarpımı

```
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
void main()
{int a[10][10],b[10][10],c[10][10], i,j,M,N,P;
cout<<"Bu program iki matrisin çarpımını bulur"<<endl;
cout<<" 1.Matris boyutlarını giriniz"<<endl;
cin>>M>>N;
for(i=0;i<M;i++)
    for (j=0;j<N;j++)
        {cout<<"<a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];cout<<endl;}
cout<<" 2.Matris boyutlarını giriniz"<<endl;
cin>>P;
for(i=0;i<N;i++)
    for (j=0;j<P;j++)
        {cout<<"<a["<<i<<"]["<<j<<"]=";
cin>>b[i][j];
cout<<endl;}
//Carpim Matrisini 0'lıyoruz.
for(i=0;i<M;i++)
    for (j=0;j<P;j++)
        c[i][j]=0;
for(i=0;i<M;i++)
    for (j=0;j<P;j++)
        for(int k=0;k<N;k++)
            c[i][j]=c[i][j]+a[i][k]*b[k][j];
//Yazdırma
for(i=0;i<M;i++)
    for (j=0;j<P;j++)
        cout<<"c["<<i<<"]["<<j<<"]="<<c[i][j]<<endl;
system("pause");
}
```

```
C:\Users\pakize\Desktop\pppp\Debug\pppp.exe
Bu program iki matrisin çarpimini bulur
1.Matris boyutlarini giriniz
3 2
a[0][0]=2
a[0][1]=3
a[1][0]=4
a[1][1]=5
a[2][0]=6
a[2][1]=1

2.Matris boyutlarini giriniz
1
b[0][0]=3
b[1][0]=4
c[0][0]=18
c[1][0]=32
c[2][0]=22
Devam etmek icin bir tuşa basin . . .
```

Ekranda girilen matrisler ve matematiksel çarpımı aşağıdaki gibidir.

$$a = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \quad c = aXb = \begin{bmatrix} 2x3 + 3x4 \\ 4x3 + 5x4 \\ 6x3 + 1x4 \end{bmatrix} = \begin{bmatrix} 18 \\ 32 \\ 22 \end{bmatrix} \text{ bulunur.}$$

Örnek: 1'den 10'a kadar sayıların çarpımını diziye yazdırın programı.

iki boyutlu dizi örneği

```
//iki boyutlu dizi örneği
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
void main()
{int d=0;int a[11][11];
int i,j;
for (i=1;i<=10;i++)
    for (j=1;j<=10;j++) a[i][j]=i*j;
//Çarpımları yaz
for (i=1;i<=10;i++)
    for (j=1;j<=10;j++)
        cout<<i<<"x"<<j<< "="<<a[i][j]<<endl;
        cout<<endl;
system("pause");}
```

```
c:\Documents and Settings\DuzceUnv\Desktop\yeni\Debug\yeni.exe
9x1=9
9x2=18
9x3=27
9x4=36
9x5=45
9x6=54
9x7=63
9x8=72
9x9=81
9x10=90
```

Örnek: Dışarıdan girilen boyutta rastgele bir iki boyutlu matris üretip, değerlerini sıralayan program.

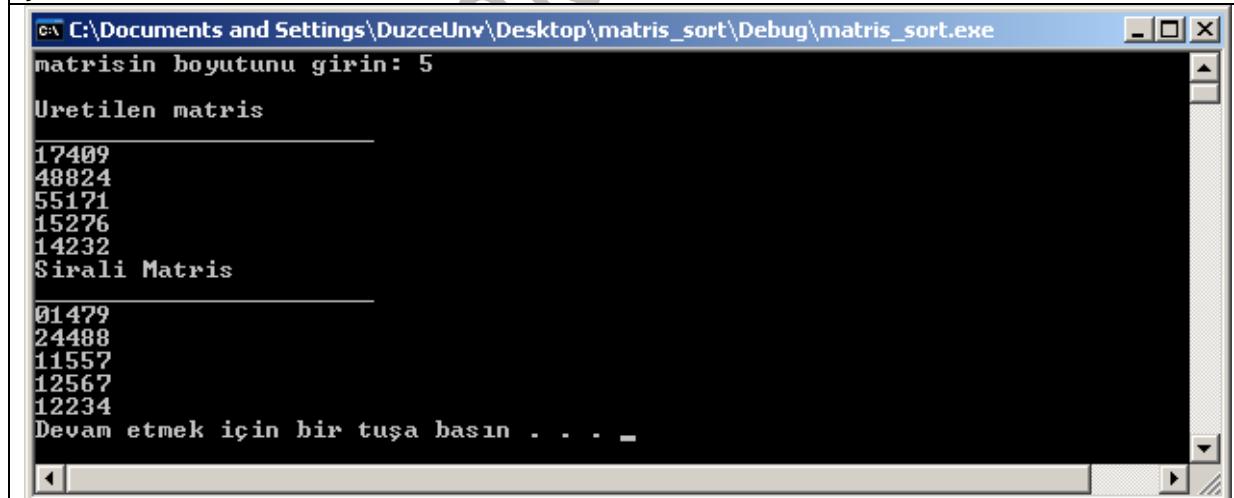
İki boyutlu matris değerlerini sıralayan program örneği

```
//28.03.2011
#include "StdAfx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{ int i,j,N;
cout << "matrisin boyutunu girin: "; cin>>N;
cout<<endl;
int A[40][40];
//Rastgele matris üretimi
    cout<<"Uretilen matris"<<endl;
    cout<<"_____ "<<endl;

    for (i=0;i<N;i++)
    {for(j=0;j<N;j++)
        {A[i][j]=rand()%10;cout <<A[i][j]; if(j==N-1) cout<<endl;}
    cout<<"Siralı Matris"<<endl;
    cout<<"_____ "<<endl;

//Rastgele matris elemanlarının sıralanması
    for (i=0;i<N;i++)
        {int k,ek;   for(j=0;j<N;j++){for (k=0;k<N;k++)
            { if (A[i][j]<A[i][k]) { ek=A[i][j]; A[i][j]=A[i][k]; A[i][k]=ek; }
            }}}

    for (i=0;i<N;i++)
    {for(j=0;j<N;j++)
        {cout <<A[i][j]; if(j==N-1) cout<<endl; }
    system("pause");
    return 0;
}
```



8.2. İki Boyutlu Karakter Dizilere İlk Değer Atama

Aşağıda indisleri verilen karakter bir dizinin elemanları görülmektedir. Bu dizinin C++'da tanımı aşağıdaki gibi yapılır.

0	1	2	3
1	f	g	h
2	i	j	K

```
char a[][4]={{'f','g','h','\0'},{'i','j','k','\0'}};
```

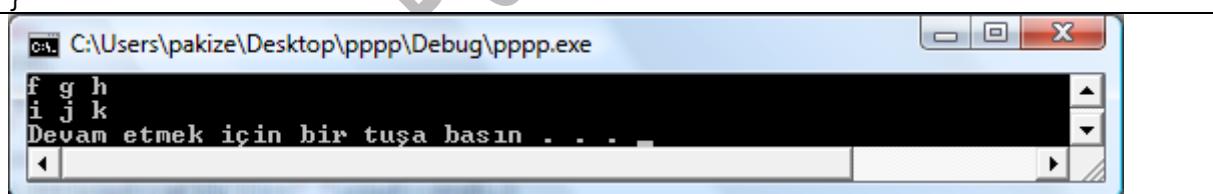
char a[][4]={"fg","ijk"}; şeklinde tanımlanır. Bu ikinci tanımda sonlandırma karakteri otomatik olarak C++ tarafından eklenir.

Örnek: Yukarıdaki karakter diziyi ekrana yazdırın program örneği

Karakter dizi örneği

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{char a[][4]={{'f','g','h','\0'},{'i','j','k','\0'}};
for(int i=0;i<2;i++)
{for (int j=0;j<3;j++)cout<<a[i][j]<<" ";cout<<endl;}
system("pause");
return 0;
}
```

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{char a[][4]={"fg","ijk"};
for(int i=0;i<2;i++)
{for (int j=0;j<3;j++)cout<<a[i][j]<<" ";cout<<endl;}
system("pause");
return 0;
}
```



Soru: Yukarıdaki programda ikinci for'un yanındaki komutlar kıvırcık parantez içine alınırsa ekran çıktısı ne olur? {cout<<a[i][j]<<" ";cout<<endl;}

Örnek: İki boyutlu karakter dizisi örneği

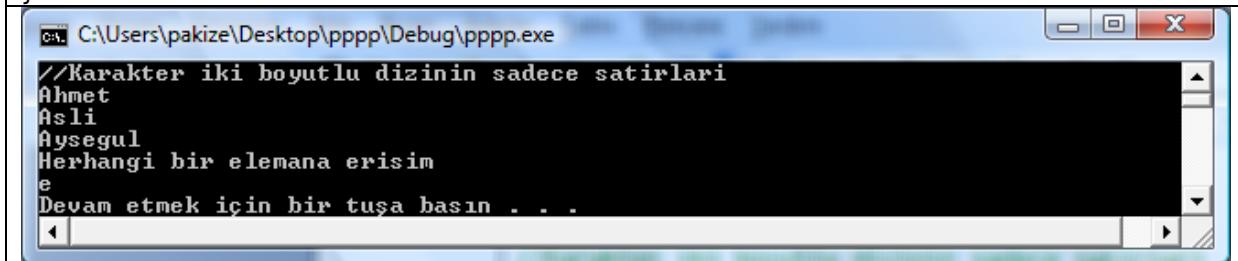
İki boyutlu karakter dizisi

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
char a[][10] = {"Ahmet","Asli","Aysegul"};
//Karakter iki boyutlu dizinin sadece satırları
cout<<"/Karakter iki boyutlu dizinin sadece satırları "<<endl;
for(int i=0;i<=2;i++)
    cout<<a[i]<<endl;
cout<<"Herhangi bir elemana erişim"<<endl;
```

```

cout<<a[0][3]<<"\t"<<a[0][8]<<"\t"<<a[0][9]<<endl;
system("pause");
return 0;
}

```



Örnek: İki boyutlu karakter dizisi örneği

İki boyutlu karakter dizisi örneği

```

#include "stdafx.h"
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
    char a[][100] ={ "Gu0967879Z8E9L!B544i6876r,d3U56nyA09." };
    char s = a[0][0];
    for(int i = 0; s != '\0'; s = a[0][++i])
    {
        if(isalpha(s))
            cout<< (char)(isupper(s) ? tolower(s) : s);
        else if(ispunct(s))
            cout << ' ';
    }
    cout<<endl;
    system("pause");
    return 0;
}

```



Soru: Bu örnek yardımcı ile bir string dizisinde(iki boyutlu karakter dizi) aranan bir harften hangi satırda kaç adet olduğunu bulan bir program yazınız.

BÖLÜM SONU SORULAR VE UYGULAMALAR

1. Bir matriste istenen bir satır elemanlarını bir diziye atan programı yazınız.
 2. Bir matrisin transpozesini alan program yazınız.
 3. Bir matriste dışarıdan girilen bir değerden büyük elemanların olduğu satırı bir küçük olduğu satırı sıfır koyan bir program yazınız.
 4. Bir matriste dışarıdan girilen bir değerden büyük eleman sayısını bulan bir program yazınız.
 5. Bir matristeki en küçük elemani ve indislerini bulan bir program yazınız.
 6. Bir matristeki elamanları bir diziye sıralı şekilde yerleştiren bir program yazınız.
 7. Katsayıları dışarıdan girilen 2 bilinmeyenli denklem sistemi çözümünü bulan bir program yazınız.
 8. Amiral Battı oyununu programlayınız.

Oyun hakkında bilgi: İki kişilik bir oyundur. Bilgisayar ve sizin aranızda oynanacak. Her iki oyuncunuda 10X10'luk bir denizi var. Bu denizde bir 5 uzunluklu, bir 4 uzunluklu, bir 3 uzunluklu, bir 2 uzunluklu gemi var. Oyun başlangıcında önce siz gemileri yerleştiriyorsunuz. Bilgisayarda kendi gemilerini yerleştiriyor. 10X10 alanında rastgele satır ve sütun değerleri girilerek rakibin(bilgisayar) gemiler batırılmaya çalışılır. Gemilerinin hepsi batan oyunu kaybeder.