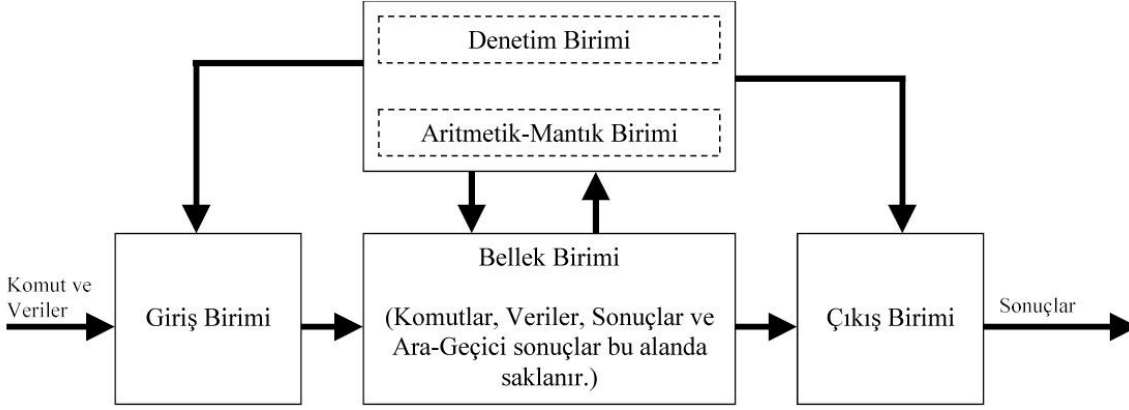


PROGRAMLAMAYA GİRİŞ I DERS NOTLARI

1. Genel Bilgiler

Bilgisayar, verileri saklayan , bunlar üzerinde çok hızlı işlem yapan ve istenen verileri sunan bir aygıttır.

Donanım (hardware) ve yazılım (software) diye iki bölüme ayrılır. Donanım bilgisayarın fiziksel bileşenleridir. Yazılım ise donanımı oluşturan bileşenlerin çalışmasını ve işlevlerini yerine getirmesini sağlayan programlardır.



Giriş Birimleri : Veri ve program girilmesini sağlar. Klavye, fare, kart okuyucu ...

Çıktı Birimleri : İstenen verilerin kullanıcıya sunulduğu ortam. Ekran, yazıcı...

Ana Bellek: Programların ve işlenen verilerin geçici olarak saklandığı birim.

Yan Bellek : Bilgilerin (veri, program) kalıcı olarak saklandığı ortamlar. Disket, disk, manyetik şerit.

Bir programcı genel olarak bu birimlerin hangilerinin ne işe yaradığını ve neleri temsil ettiğini bilmelidir. Özellikle Aritmetik-Mantık işlem birimi ve Denetim birimi bir bilgisayarın beynini oluşturduğu için nasıl davrandığını iyi bilmek zorundadır.

Problem Nedir?

Bir işlemin, otomasyonun yada bilimsel hesaplamanın bilgisayarla çözülmesi fikrinin ortaya çıkmasına problem denir. Bu tip fikirlerde insanların bu sorunları beyinle çözmeleri ya imkansızdır ya da çok zor ve zaman alıcıdır. Bu tip bir sorunu bilgisayarla çözebilme fikrinin ortaya çıkması bir bilgisayar probleminin ortaya çıkmasına neden olmuştur. Bazen de bir işletme veya yönetimin otomasyonunu sağlamak amacı ile bu tip problemler tanımlanır.

Problem Çözümü

Problemi Çözebilmek için öncelikle sorunun çok net olarak programcı tarafından anlaşılmış olması gerekir. Tüm ihtiyaçlar ve istekler belirlenmelidir. Gerekliyse bu işlem için birebir görüşmeler planlanmalı ve bu görüşmeler gerçekleştirilmelidir. Problemin Çözümüne ilişkin zihinsel alıştırmalar yapılır. Bu alıştırmaların Bilgisayar çözümüne yakın olması hedeflenmelidir. Bir sorunun tabii ki birden fazla çözümü olabilir. Bu durumda bilgisayar ile en uygun çözüm seçilmelidir. Çünkü bazen pratik çözümler bilgisayarlar için uygun olmayabilir.

Oluşturulan Çözüm Algoritma dediğimiz adımlarla ifade edilmelidir.

Bu algoritmanın daha anlaşılabilir olması için Akış Çizgesi oluşturulmalıdır.

Uygun bir programlama dili seçilmeli ve oluşturulan algoritma ve akış çizgesi bu programlama dili aracılığı ile bilgisayar ortamına aktarılmalıdır.

Oluşturulan program bir takım verilerle ve mümkünse gerçek ortamında test edilir. Oluşabilecek sorunlar ilgili kısımlar tekrar gözden geçirilerek düzeltilir. Bu adımlar defalarca gerçekleştirilmek zorunda kalınabilir.

İster bilgisayarla ister bilgisayarsız soru çözmek için belirli bir yol vardır. Ancak bu yol ile sağlıklı bir çözüme ulaşılabilir. Bilgisayar kullanarak soru çözmek için sonuca giden yolun tam olarak belirlenmesi gerekir. Doğru bir yol izleyebilmek için, çıkılan ve ulaşılan yer tanımlanmalıdır. Aynı soru için değişik çözüm yolları geliştirilebilir. Eğer bilgisayara verilen çözüm yanlışsa, çıkan sonuç yanlış çözüm doğru ise çıkan sonuç da doğrudur.

Problem Çözme Adımları :

Bilgisayar ortamında bir problem çözülürken aşağıdaki adımlara dikkat edilmelidir.

a-) Problemi Tanımlama: Her şeyden önce çözülecek problem tam olarak anlaşılmalıdır. Yanlış anlaşılmuş bir problemin çözümü yanlış olacak ve istenileni vermeyecektir. Bu adımda yapılacak en ufak bir hata daha sonraki adımların yeni baştan yapılmasını gerektirebilir. Sorunun tanımı yapılırken var olan bilgiler, anlamları ve birbirleri ile ilişkileri tanımlanmalıdır. Daha sonra istenenler belirlenmeli ve bunların var olan bilgiler ile ilişkileri öğrenilmelidir. Son olarak yapılacak işlemler belirlenir. Mümkün ise örnek veriler ile elde edilen sonuçlar değerlendirilmelidir.

b-) Algoritma Geliştirme: Algoritma bir sorunun çözümü için izlenecek yolun tanımıdır. Kısaca algoritma mevcut bilgilerden istenilenlere erişme yöntemidir. Problem tanımını tam olarak yaptıktan sonra, çözüm için yol aramak gerekir. Genellikle bir sorunun birden fazla çözüm yolu olabilir. Bunlardan en uygunu seçilmeye çalışılır. Problem ne kadar karışık olursa olsun, alt birimlere bölünür. Her birimin çözümü ayrı, ayrı yapılır. Bu yapılırken birimler arası ilişki sürekli olarak korunur.

c-) Girdi ve Çıktı Biçimi Belirleme: Sonuçların dış ortama, dolayısıyla insana aktarımı düzgün bir biçimde yapılmalıdır. Programcı program çıktısı olarak almak istediği dökümün biçimini tasarlar. Bir döküm biçimi tasarlanırken anlaşılır ve kullanılabilir olmasına özen gösterilmelidir. Genellikle programa, çözdüğü soruna ilişkin bazı verilerin dışarıdan verilmesi gerekir. Örneğin bir denklem takımının kökleri bulunacaksa, ilgili katsayıların programa verilmesi gibi.

d-) Akış Şemasını Çizme: Akış şeması belirli bir işin yapılabilmesi için, basit işlemlerle şema halinde gösterilmesidir. Kısaca algoritmanın şemalarla gösterilmesidir. Algoritma geliştirildikten sonra, daha iyi anlaşılabilir olması ve programlama dillerine aktarımı daha kolay olması nedeniyle, akış şeması haline getirilir. Böylece sorunun çözüm basamakları, birbirleri ile ilişkileri ve bilgi akışı daha kolay görülebilir ve yanlışlıklar düzeltilebilir.

e-) Kodlama: Akış şemaları çizildikten sonra, sorunu yapısına uygun bir programlama dili seçilir. Bu dil ile akış şemaları dilin kurallarına uygun olarak bilgisayarın anlayabileceği duruma getirilir.

f-) Programı Sınama: Program yazıldıktan sonra, sonuçları daha önceden bilinen veriler girilerek, eldeki sonuçlarla çıkan sonuçlar karşılaştırılır. Programın doğru çalışıp çalışmadığı sınanır.

Program Nedir?

Problem Çözümü kısmında anlatılan adımlar uygulandıktan sonra ortaya çıkan ve sorunumuzu bilgisayar ortamında çözen ürüne Program denir. Bazı durumlarda bu ürüne yazılım denebilir.

Programlama Nedir?

Problem Çözümünde anlatılan adımların tümüne birden programlama denilebilir. Ancak son paragrafta anlatılan adıma kısaca test aşaması da denir. Çoğunlukla çok iyi tanımlanmış bir sorunun çözümüne dair adımlar ile çözümün oluşturulup bunun bir programlama dili ile bilgisayar ortamına aktarılması Programlama diye adlandırılabilir.

Algoritma Nedir?

Bir sorunu çözebilmek için gerekli olan sıralı mantıksal adımların tümüne denir. Doğal dille yazılabileceği için fazlaca formal değildir. Bir algoritma için aşağıdaki ifadelerin mutlaka doğrulanması gereklidir.

- Her adım son derece belirleyici olmalıdır.
- Hiç bir şey şansa bağlı olmamalıdır.
- Belirli bir sayıda adım sonunda algoritma sonlanmalıdır.
- Algoritmalar karşılaşılabilecek tüm ihtimalleri ele alabilecek kadar genel olmalıdır.

Akış Çizelgesi Nedir?

Bir algoritmanın daha görsel gösterimidir. Çizgiler, Dörtgen, daire vb. geometrik şekillerle algoritmanın gösterilmesini sağlar. Doğal dille yazılmadığı için daha formal olduğu düşünülebilir.

Programlama Dili Nedir?

Bir Problemin Algoritmik çözümünün Bilgisayara anlatılmasını sağlayan,son derece sıkı-sıkıya kuralları bulunan kurallar dizisidir.



Derleyici Nedir?

Bir programlama dili ile bilgisayara aktarılan programın bilgisayarın anlayabileceği Makine Diline çevirmeyi sağlayan ve yazılan programda söz dizim hatalarının olup olmadığını bulan olup olmadığını bulan yazılımlardır. Her Programlama dili için bir derleyici olması gerekmektedir.

Yorumlayıcı Nedir?

Derleyici gibi çalışan ancak yazılmış programları o anda Makine diline çeviren yazılımlardır. Bu tür bir yazılımda Programın Makine dili ile oluşturulmuş kısmı bilgisayarda tutulmaz. Programın her çalıştırılmasında her adım için Makine dili karşılıkları oluşturulur ve çalıştırılır.

Programlama Dili Seçimi : Çözümün netleşmesinden sonra yapılacak işlemleri kolay bir şekilde bilgisayar ortamına aktaracak dilin seçilmesidir. Önemli olan bu dilin özelliklerinin programcı tarafından iyi bilinmesidir.

Programın Yazılması : Seçilen Programlama dilinin kuralları kullanılarak program yazılmaya başlanır. bu amaçla çoğunlukla sade bir metin editörü kullanılır. Bazı durumlarda Syntax highlighting denilen bir özelliğe sahip olan daha akıllı editörler de kullanılabilir. Bazen de editör ile Programlama dilinin derleyicisinin, bağlayıcısının hatta hata ayıklayıcısının iç içe bulunduğu IDE (Integrated Development Environment) denilen türde derleyiciler kullanılır.

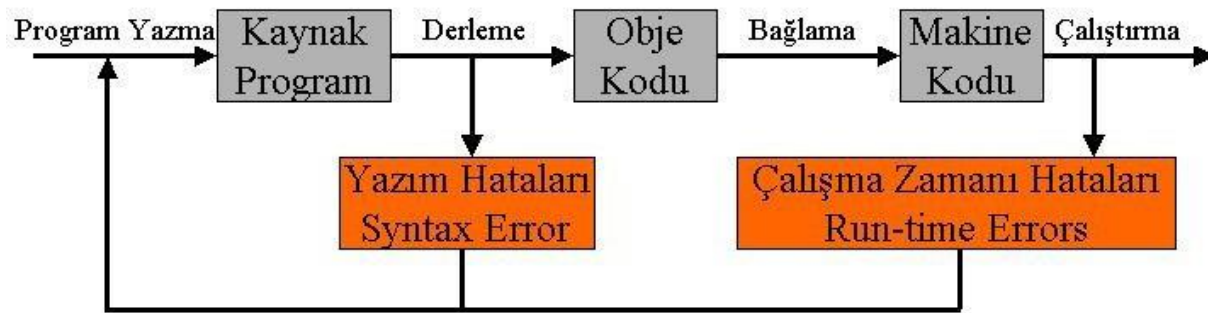
Derleme : Programlama Dili ile yazılmış programın yazım hatalarının olup olmadığının kontrol edilmesini ve ara kod olarak Obje kodun üretilmesini sağlama adıdır.

Bağlama : Derlenmiş ara kod diğer kütüphane ve parça programlarla birleştirilerek Makine dilinde programın oluşturulması adıdır. Ancak bazı IDE ortamlarda ve derleyicilerde Derleme ve Bağlama bir bütündür ve beraberce halledilirler. Programcının ayrıca bir bağlama işlemi yapması gerekmez işlemi yapması gerekmez.

Çalıştırma : Oluşturulan Makine dili Programının çalıştırılması adıdır. Yukarıdaki adımların hepsi yolunda gittiyse program sorunsuz olarak çalışabilmelidir.

Test : Programın Mantıksal olarak test edilmesini sağlar ve içerik olarak her ihtimal için doğru sonuçlar üretilip üretilmediğini kontrol etmenizi sağlar.

Hata Yakalama ve Ayıklama:



Bir Programın bilgisayar başında geçen geliştirme süreci yukarıdaki gibidir. Bu çizimde kırmızı-turuncu renkle gösterilen kısımlar hata durumlarını göstermektedir.

Syntax Error : Yazılan programda programlama dili kurallarına aykırı bir takım ifadelerden dolayı karşılaşılabilecek hatalardır. Düzeltmesi son derece basit hatalardır. Hatanın bulunduğu satır derleyici tarafından rapor edilir. Hatta bazı derleyiciler hatanın ne olduğunu ve nasıl düzeltilmesi gerektiğini dahi bildirebilirler. Bazen Syntax Error tipi hataları Bağlama zamanında da ortaya çıkabilir. Eğer bir derlemede Syntax Error alındı ise obje kod üretilmemiştir demektir.

Soru: Bir Derleyici hatanın nasıl düzeltileceğini bildirebildiğine göre kendisi niçin düzeltmemektedir?

Run-time Error : Programın çalıştırılması sırasında karşılaşılan hatalardır. Programcının ele almadığı bir takım aykırı durumlar ortaya çıktığında programın işletim sistemi tarafından kesilmesi ile ortaya

çıkar. Bu tip hatalarda hata mesajı çoğunlukla çalışan işletim sisteminin dili ile verilir. Eğer bu tip hataları kullanıcı ele almışsa, program programcının vereceği mesajlarla ve uygun şekilde sonlandırılabilir. Bu tip hataların nerelerde ve hangi şartlarda ortaya çıkabileceğini bazen kestirmek zor olabilir. Çoğunlukla işletim sistemi ve donanım kaynakları ile ilgili sorunlarda bu tip hatalar ortaya çıkar demiştik. Örneğin olamayan bir dosya açmaya çalışmak, var olan bir dosyanın üzerine yazmaya çalışmak, olmayan bir bellek kaynağından bellek ayırtmaya çalışmak, olmayan bir donanıma ulaşmaya çalışmak vs. vs. vs.

Logical Error : Karşılaşılabileceğiniz en tehlikeli hatadır. Programlama mantığında bir takım şeylerin yanlış düşünülmesinden kaynaklanır. Hata test aşamasında ortaya çıkar. Hesaplanması gereken veya bulunması gereken değerlerin eksik veya yanlış hesaplanması ile tespit edilir. Bu sorunun giderilebilmesi için Tasarım hatta çözümleme aşamasına geri dönülmesi gerekebilir. Bazen bu hatanın nereden kaynaklandığını bulabilmek çok zor olmaktadır.

Bug : Logical Error diyebileceğimiz Mantıksal hatalara verilen adlar bug yani böcek diye de tanımlanmış olabilir. Bu tip hatalar eğer çok net değil ve zamanla ortaya çıkabiliyor ise veya nedeni çok net olarak anlaşılamamışsa bug diye adlandırılır. Gerek serbest yazılım gerek ticari yazılımların tümünde bug dediğimiz mantıksal hatalar bulunur. Çünkü hatasız program yazabilmek çok zordur. İlk seferde yazılan bir programın tamamen hatasız olmasını beklemek son derece hatalıdır. Günümüzde en meşhur yazılım firmaları bile yazılımlarında bug olduğunu kabul eder ve zaman zaman bu bugları giderebilmek için ya yazılımlarına yama yazılımı üretirler yada o yazılımın yeni bir versiyonunu piyasaya sürerler.

Debug : Mantıksal hataları giderebilmek ve yazılımdaki bug'ları bulabilmek için yapılan işlemin adıdır. Genellikle yazılan programın adım adım ve denetim altında çalıştırılmasıdır. Programın her adımında ilgili değişkenlerin hangi değere sahip olduğunu görmeyi sağlar ve anormal bir durumu daha kolay izleyip bulmanızı sağlar. Bu işlemi gerçekleştirebilmek için bazı IDE ortamlarında debugger dediğimiz yardımcı seçenekler kullanılır.

Örnek bir Algoritma

Örneğimiz bir insanın evden çıkıp işe giderken izleyeceği yolu ve işyerine girişinde ilk yapacaklarını tanımlamaktadır.

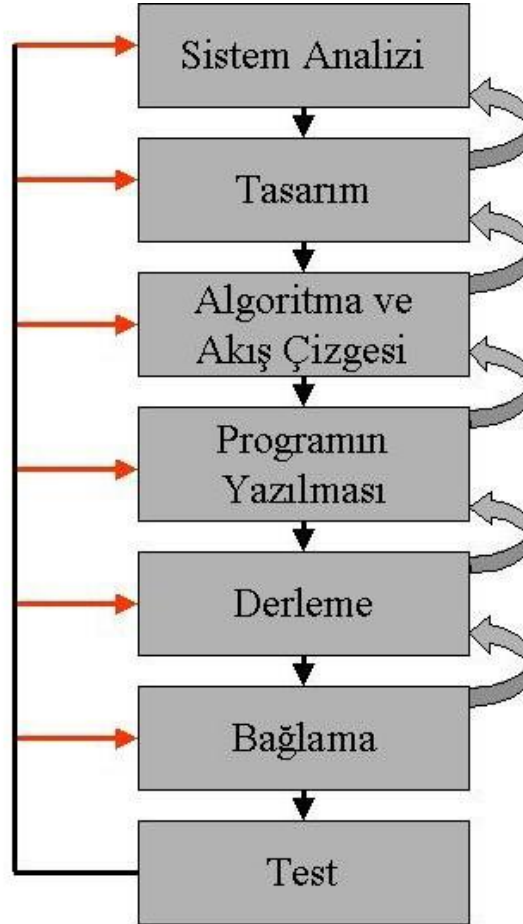
Çözüm 1:

Evden dışarıya çık
 Otobüs durağına yürü
 Durakta gideceğin yöndeki otobüsü bekle
 Otobüsün geldiğinde otobüse bin
 Biletini bilet kumbarasına at
 İneceğin yere yakınlaştığında arkaya yürü
 İneceğini belirten ikaz lambasına bas
 Otobüs durunca in
 İşyerine doğru yürü
 İş yeri giriş kapısından içeriye gir
 Mesai arkadaşlarınla selamlaş
 İş giysini giy
 İşini yapmaya başla.

Şimdi algoritma hazırlama işlemini daha ayrıntılı olarak inceleyelim. **Cünkü programlamanın en önemli kısmı algoritma hazırlayabilmektir. Algoritma hazırlandıktan sonra hazırlanan algoritmanın herhangi bir programlama dilinde kodlanması işin en basit kısmıdır. Bu yüzden kullanılan dili hiçbir şeyi değıştirmez. Yani burada önemli olan programlama dili değil problemin çözümü için algoritma geliştirebilmektir. Bu yüzden kullanılan programlama dilinin eski yada yeni bir programlama dili olması hiç önemli değildir.**

Yazılım Geliştirme

Yazılım Geliştirilirken Bir Programcı ve Yazılım Gurubunun takip edeceği adımlar şu şekildedir.



Bu çizgeden anlaşılacağı gibi adımlardan birinde bir sorunla karşılaşırsa bu sorunu çözebilmek için bir önceki adıma geri dönmek gerekecektir. Bu geri dönüş bazen bir kaç adım olabilir.

Sistem Analizi : Sorunun çözülebilmesi için tamamen anlaşılmasını sağlayan çalışmalardır.

Tasarım : İsteklerle ilgili olarak belirlenen bir takım çözümlerin tanımlanmasıdır.

Programlama Stili : Her yiğidin yoğurt yiyişi farklıdır. Aynı şekilde her programcı programındaki mantığı farklı kurar bu her programcının kendine özgün bir stili var anlamına gelir. Ancak bunun yanında Her programcının programın sağlığı bakımından dikkat etmesi gereken şeyler vardır. Örneğin kodlar açık olmalıdır. Kullanılan değışkenler kullanıldıkları amacı anlatır tarzda isimlendirilmelidir. Program içi dokümantasyona mutlaka önem verilmelidir.

Algoritma : Çözümün adımlarla ifade edilmesidir.

Akış Çizgesi : Algoritmanın şekillerle ifade edilmesidir.

Programlama Dili Seçimi : Çözümün netleşmesinden sonra yapılacak işlemleri kolay bir şekilde bilgisayar ortamına aktaracak dilin seçilmesidir.Önemli olan bu dilin özelliklerinin programcı tarafından iyi bilinmesidir.

Programın Yazılması : Seçilen Programlama dilinin kuralları kullanılarak program yazılmaya başlanır. bu amaçla çoğunlukla sade bir metin editörü kullanılır. Bazı durumlarda Syntax highlighting denilen bir özelliğe sahip olan daha akıllı editörler de kullanılabilir. Bazen de editör ile Programlama dilinin derleyicisinin, bağlayıcısının hatta hata ayıklayıcısının iç içe bulunduğu IDE (Integrated Development Environment) denilen türde derleyiciler kullanılır.

Derleme : Programlama Dili ile yazılmış programın yazım hatalarının olup olmadığının kontrol edilmesini ve ara kod olarak Obje kodun üretilmesini sağlama adıdır.

Bağlama : Derlenmiş ara kod diğer kütüphane ve parça programlarla birleştirilerek Makine dilinde programın oluşturulması adıdır. Ancak bazı IDE ortamlarda ve derleyicilerde Derleme ve Bağlama bir bütündür ve beraberce halledilirler. Programcının ayrıca bir bağlama işlemi yapması gerekmez işlemi yapması gerekmez.

Çalıştırma : Oluşturulan Makine dili Programının çalıştırılması adıdır.Yukarıdaki adımların hepsi yolunda gittiyse program sorunsuz olarak çalışabilmelidir.

Test : Programın Mantıksal olarak test edilmesini sağlar ve içerik olarak her ihtimal için doğru sonuçlar üretilip üretilmediğini kontrol etmenizi sağlar.

Sayı Sistemleri

Bir Bilgisayar sisteminde tüm bilgi kayıtları ve işlemleri elektriksel devreler üzerinden gerçekleştiği için tek bilinen gerçek elektrik akımının varlığı veya yokluğudur. Bu da matematiksel ve mantıksal olarak ikili sayı sistemine karşılık gelir. Çoğunlukla ikili sayı sistemindeki 0 değeri elektrik olmadığını 1 değeri ise bir elektriksel gerilimin olduğunu anlatır. Bu iki sayısal değer (0 ile 1) ikili sayı sisteminin rakamlarıdır. Ve bilgisayarda oluşan tüm değer ve sonuçlar gerçekte bu rakamlar ile anlatılabilirler. Ancak bizim bu sayısal değerleri anlamamız zor olduğu için sayısal olarak onluk sayı sistemini kullanırız.

Tabandan Tabana Çevrim

Böyle olunca sayıların gerektiği durumlarda tabandan tabana çevrilebilmesi gereklidir. İlköğrenim düzeyinde görmüş olabileceğiniz yöntemlere burada bir değinmekte fayda bulunmaktadır. Bu rakamların her biri bilgisayar da bit denilen alanlarda tutulmaktadır. İkili Sayı isteminden onlu sayı sistemine çevrim Elimizdeki ikili sayının en sağındaki basamak sıfıncı basamak olmak kaydıyla tüm basamaklarımız sola doğru numaralandırılır. Sonra her basamaktaki sayısal değeri 2^{basamak} değeri ile çarpıp ve bulunan tüm değerleri toplarız.

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	1

$$=1*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 1*2^4 + 0*2^5 + 0*2^6 + 1*2^7$$

$$=1 + 2 + 0 + 8 + 16 + 0 + 0 + 128$$

$$=155$$

Onlu Sayı isteminden ikili sayı sistemine çevrim

Eldeki onlu sayı sürekli 2 değerine bölünerek işlem yapılır. Bölme işlemi en son bölümün 0 olduğu noktaya kadar devam eder. Elde edilen bölme tablosunda en son kalanlar sondan başa doğru yan yana yazılır ve sayının ikilik tabandaki karşılığı bulunmuş olur.

Örneğin elimizde 156 gibi sayısal bir değer olsun.

İşlem	Bölüm	Kalan
155 / 2	77	1
77 / 2	38	1
38 / 2	19	0
19 / 2	9	1
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

Sonuç Kalan sütunundaki değerlerin aşağıdan yukarı dizilmesi ile 1 0 0 1 1 0 1 1 olarak elde edilir.

Diğer Sayı Sistemleri

0..X şeklinde bir dizi rakama sahip olan sayı sistemleri X+1’li sayı sistemi olarak anılır. Örneğin 0..7 arasında rakamlarla ifade edilen sayı sistemi 8’li (Octal) sayı sistemidir. Bunun gibi bir programcının bilmesi gerekebilecek 16’lı (hexadecimal) sayı sistemi vardır. Bu sistemde sayılar 0..9’a kadar gider ve sonrasında A,B,C,D,E,F gibi rakamları da 10,11,12,13,14,15 gibi değerleri ifade etmek için kullanılır. Bu sayı sistemleri haricinde 3’lü, 5’li vb sistemler bulunabilir/bulunur. Ancak bir bilgisayar programcısı ikili, sekizli, onlu, onaltılık sistemleri kullanır ve bu sistemlerden haberdardır.

$$(1111100101110011)_2 = (F973)_{16} = (174563)_8$$

2. Algoritma Hazırlanması:

Algoritmanın bir sorunun çözümü için izlenecek yolun tanımı olduğunu belirtmiştik. Bu yolun açıklanabilmesi için, algoritmada kullanılan bazı tanım ve kurallar vardır. Şimdi bu tanım ve kuralları inceleyelim.

2.1. Değişken Kavramı:

Farklı zamanlarda farklı değerler alabilen bilgi sahalarına verilen sembolik adlardır. Bilgisayar işlem yaparken RAM belleği(geçici bellek) kullanır. İşte program yazılırken programcının Ram belleği kullanmasını sağlayan değişkenlerdir. Değişkenler Ram bellekte tahsis edilmiş odacıklar olarak düşünülebilir. Yani bir değişken tanımlandığında ram bellekte bir odacık (bir bölüm) açılır ve bu bölüme değişken ismiyle ulaşılır. Program içinde kullanılacak olan değişkenler problemin tanımı ve girdi-çıkıtlı belirleme aşamalarında belirlenmelidir.

Örneğin klavyeden girilen iki sayının toplamını bulan program yazılırken 3 tane değişken tanımlanmalıdır. Çünkü klavyeden 2 tane sayı girilecek ve bu sayılar toplanarak 3. bir değişkene aktarılacaktır.

Sabit Nedir?

Programın her yerinde aynı değeri ifade eden değerlerdir. Değişkenlerde olduğu gibi sabitler de tür kavramına sahiptir. Yazılış tarzına bakarak sabitin türünü anlamak mümkündür.

Soru: Klavyeden girilen 3 sayının aritmetik ortalaması bulunurken kaç değişken tanımlanmalıdır.

2.2. Aktarma Deyimi:

Aktarma deyimi yada operatörü değişkenlere değer aktarmak için kullanılır. $A=5$ yada $A=A+1$ şeklindeki bir yazılımda “=” sembolü aktarma deyimi adını alır. Aktarma deyiminin sağ tarafındaki değer yada matematiksel ifadenin sonucu, sol tarafındaki değişkene aktarılır. Aktarma yapılırken değişkenin aldığı bir önceki değer kaybolur. Bu işlem matematiksel mantıkla karıştırılmamalıdır. Matematikte $A=A+1$ yanlış olduğu halde, bilgisayar mantığında doğrudur.

Sayı1=9

Sayı2=6

Toplam=Sayı1+Sayı2

Toplam=Toplam*2

Yandaki işlemlerin sonucunda bellekteki değişkenlerin değerleri şu şekilde değişir.

Sayı1	Sayı2	Toplam
9	6	15 30

RAM BELLEK

Soru 1:

Sayı1=1

Sayı2=1

Sayı3=Sayı1+Sayı2

Sayı4=Sayı3+Sayı2

Sayı5=Sayı4+Sayı3

Yukarıdaki aktarma işlemlerinin sonucunda değişkenlerin değişimini şema halinde gösteriniz.

Soru2:

Vize=60

Final=70

Toplam=(Vize*40/100)+(Final*60/100)

Ortalama=Toplam/2

Yukarıdaki aktarma işlemlerinin sonucunda değişkenlerin değişimini şema halinde gösteriniz.

2.3. Matematiksel Mantık ve Karar Sembolleri:

Algoritmada kullanılan karar sembolleri aşağıdaki tabloda belirtilmiştir.

-	+	*	/	=
Çıkarma	Toplama	Çarpma	Bölme	Aktarma
<>, !=	<	>	<=	>=
Eşit değil	Küçüktür	Büyüktür	Küçük eşit	Büyük eşit

Örnekler:

Dyili=1977

Yas=2003-1977

1.Sayi=0

2.Eğer Sayı<0 ise Yaz “Sayı Negatif”

3.Eğer Sayı>0 ise Yaz “Sayı Pozitif”

4.Eğer Sayı==0 ise Yaz “Sayı Sıfırdır”

Yandaki örnekte Sayı değişkenine -5 değeri aktarılmıştır. Diğer satırlarda ise Sayı değişkeninin içeriğine bakılarak bir mesaj verilmektedir. Bu örnekte yalnızca 2. satırdaki şart sağlanır ve ekrana “Sayı Negatiftir” mesajı yazılır.

Bir veya birden fazla koşul bazı bağlaçlarla bir araya getirilerek daha karmaşık sorular sorulabilir. Örneğin yas (kişinin yaşı) değişkeninin içeriği kontrol edilmek istendiğinde şöyle bir şart cümlesi kullanılabilir:

Yas=-5

Eğer yas<0 Veya Yas==0 ise Yaz “Yanlış Değer Girildi”

Algoritmada kullanılabilecek bağlaçlar VE, VEYA, DEĞİL bağlaçlarıdır.

VE Bağlacı: Ve bağlacı ile söylenmek istenen her iki koşulun da sağlanmasıdır. VE bağlacı ile bağlanmış önermelerden en az birinin yanlış olması sonucu yanlış yapar.

VEYA Bağlacı: VEYA bağlacı ile bağlanan koşullardan en az birisi doğru ise sonuç doğrudur. İki'den fazla önermeler için, önermelerden en az birinin doğru olması sonucu doğru yapar.

DEĞİL Bağlacı: DEĞİL bağlacı doğruyu yanlış, yanlış doğru yapar. DEĞİL tek bir önerme veya koşul üzerinde uygulanır. VE, VEYA ise iki önerme veya koşul üzerinde uygulanır. **Doğru=1** ve **Yanlış=0** tanımıyla, aşağıdaki tabloda bağlaçların x ve y'nin alacağı değerlere göre sonuçları gösterilmiştir.

X	Y	X VE Y	X VEYA Z	Z	DEĞİL Z
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1		
1	1	1	1		

Örnek:

DEĞİL A=1 **VE** B=C **VEYA** (D=2 **VE** ADI="AHMET") önermesini A=5, B=4, C=4, D=3 ve ADI="AHMET" için bulalım.

DEĞİL A=1 **VE** B=C **VEYA** (D=2 **VE** ADI="AHMET")

1 **VE** 1 **VEYA** (0 **VE** 1)

1 **VEYA** 0

1

2.4. Matematiksel İşlemler:

Matematiksel işlemleri algoritmada aynen kullanamayız. Bilgisayar mantığına göre matematiksel ifadelerin yeniden yazılmaları gerekir. Algoritmada işlem öncelik sırası kuralları aşağıda verilmiştir. Parantez kullanılarak işlem öncelik sıraları değiştirilir. İç içe kullanılan parantezlerde öncelik en içtekindedir. Aynı işlem önceliğine sahip elemanlarda işlem soldan sağa doğrudur.

İşlem öncelik sırası kuralları			
Sıra	Tanım	Matematik	Bilgisayar
1	Parantezler	(())	(())
2	Üs Almak	a^n	a^n
3	Çarpma ve Bölme	ab a/b	$a*b$ a/b
4	Toplama ve Çıkarma	$a+b$ $a-b$	$a+b$ $a-b$





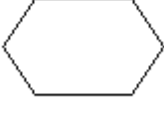

Örnek:

$$(C*D/(A*D))+B+C*D/A \longrightarrow \frac{CD}{AD} + B + \frac{CD}{A}$$

$$C+B*A/A-B^C*(B-C)^B \longrightarrow C + \frac{BA}{A} - B^C (B-C)^B$$

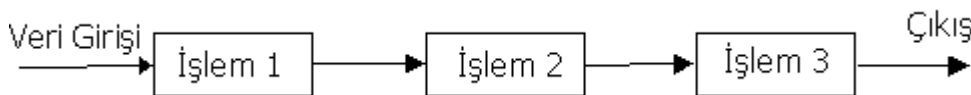
2. Akış Şeması Hazırlama:Geliştirilecek olan yazılımın genel yapısının şematik gösterimine akış şeması veya blok diyagramı adı verilir. Akış diyagramları, yazılımı oluşturacak program parçalarını ve bu parçaların birbirleri ile olan ilişkilerini belirler. ***Bir bilgisayar programının oluşturulmasında akış diyagramlarının hazırlanması, algoritma oluşturma aşamasından sonra gelmektedir.*** Bilgisayar programının oluşturulması sırasında algoritma aşaması atlanarak, doğrudan akış diyagramlarının hazırlanmasına başlanabilir. Programlama tekniğinde önemli ölçüde yol almış kişiler bu aşamayı da atlayarak direkt olarak programın yazımına geçebilirler. Akış şemalarının algoritmadan farkı, adımların simgeler şeklinde kutular içinde yazılmış olması ve adımlar arasındaki ilişkilerin (iş akışı) oklar ile gösterilmesidir. Akış şemalarında kullanılan semboller, anlamları ve kullanım amaçları aşağıdaki tabloda verilmiştir.

Tablo 1. İş akış şemalarında kullanılan semboller ve anlamları

Simge	Simgenin Adı ve Anlamı
	Elips Akış şemasının başlangıç ve bitiş yerlerini gösterir. Başlangıç simgesinden çıkış oku vardır. Bitiş simgesinde giriş oku vardır.
	Paralel Kenar: Programa veri girişi ve programdan elde edilen sonuçların çıkış işlemlerini gösterir.(Oku, Yaz)
	Dikdörtgen Aritmetik işlemler ve değişik atama işlemlerinin temsil edilmesi için kullanılır.(A=A+1, Final=100 vb..)
	Eşkenar Dörtgen Bir karar verme işlemi temsil eder. (Eğer sayı<0 ise Yaz "Sayı negatif" vb...)
	Altıgen Program içinde belirli blokların ard arda tekrar edileceğini gösterir.(Döngü kurmak için kullanılır)
	Oklar Diyagramın akış yönünü ,yani her hangi bir adımdaki işlem tamamlandıktan sonra hangi adıma gidileceğini gösterir.

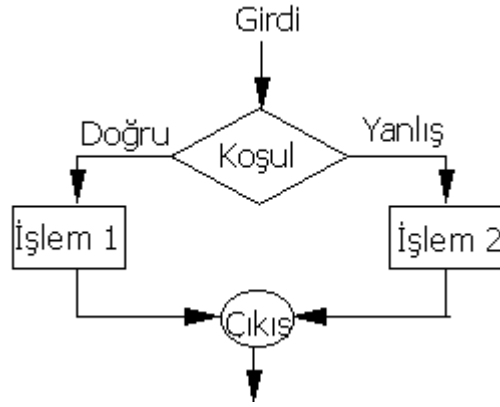
Ayrıntılı bir akış şeması, yazılımı oluşturan işlemleri ve ilişkilerini en küçük detayına kadar belirler.

Bir bilgisayar programının geliştirilmesinde kullanılan programlama dili ne olursa olsun bu programların akış diyagramlarında genel olarak yalnız üç basit mantıksal yapı kullanılır. Bu mantıksal yapılardan en basiti sıralı yapıdır. Sıralı yapı, hazırlanacak programdaki her işlemin mantık sırasına göre nerede yer alması gerektiğini vurgular. Bu yapı sona erinceye kadar ikinci bir işlem başlayamaz.



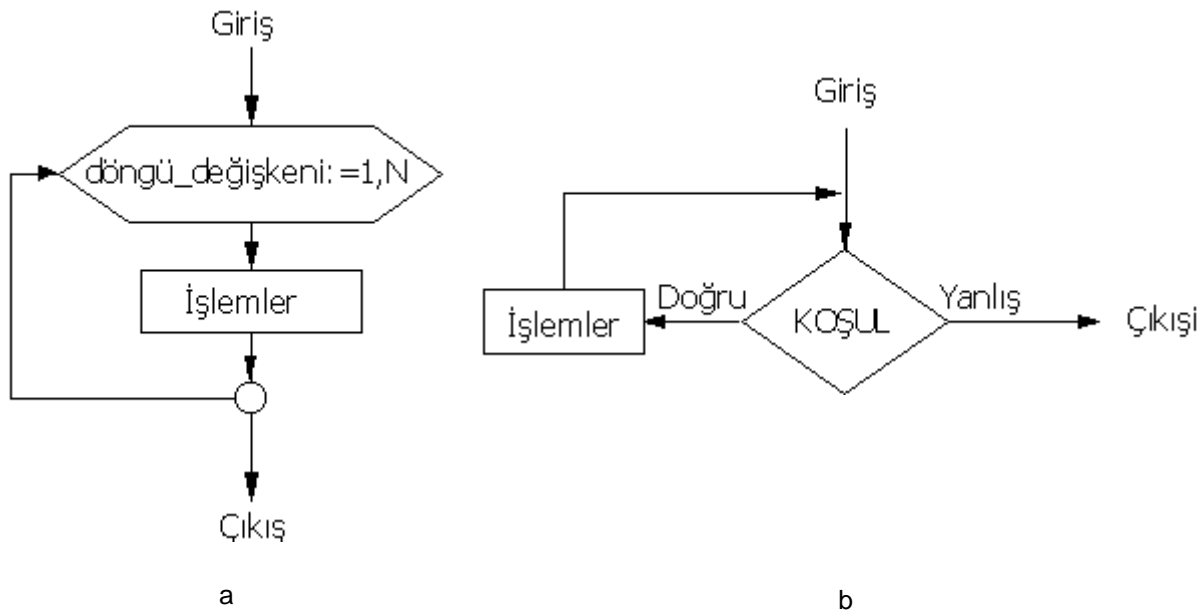
Şekil 1.2 Sıralı Yapı

Mantıksal yapılardan ikincisi Karar Verme yapısıdır (Şekil 1.3). Programlama sırasında If...Then... Else (Eğer.... <şart>İse....) yapısı ile tanıyacağımız bu mantıksal yapılar, birden fazla sıralı yapı seçeneğini kapsayan modüllerde, hangi şartlarda hangi sıralı yapının seçileceğini belirler.



Şekil 1.3 Karar Verme Yapısı

Üçüncü mantıksal yapı çeşidini tekrarlı yapılar oluşturmaktadır. Yani döngü oluşturmak için kullanılan yapıdır. Döngüler aynı işlemin bir çok kez yapılmasını sağlar. Söz konusu üç değişik yapı, değişik kombinasyonlarda kullanılarak istenilen işlevleri yerine getirecek programlar hazırlanabilir. Programların bu üç basit yapı ile sınırlandırılması program modüllerinin daha kolay tasarlanmasını sağlar.



Şekil 1.4. Tekrarlı Yapı

ÖRNEK ALGORİTMA VE AKIŞ ŞEMALARI

Örnek 1: Klavyeden girilen 2 sayının toplamını bulan programın algoritma ve akış şemasını yapınız.

Bu problemi çözerken yapmamız gereken ilk iş problemi iyice anlamaktır. Problemi çözümlersek yapılacak olan işlemler şunlardır:

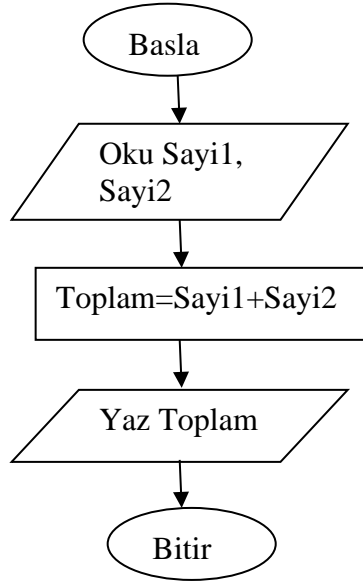
1. Klavyeden 2 adet sayı girilecek. O zaman bellekte 2 odacık açmalıyız. Yani veri girişi için 2 tane değişken kullanmak zorundayız.
2. Klavyeden girilen ve ram belleğe aktarılan bu iki değişken toplanacak. Toplam sonucunu yine ram belleğe aktarılmalıdır. Bu yüzden ram bellekte bir odacık daha açmalıyız.

3. Bulduğumuz toplam sonucunu ekrana yazdırmalıyız. Aksi taktirde toplam sonucu sadece ram bellekte bulunur ve sonucu kullanıcı göremez. Şimdi bu açıklamalar ışığında algoritma ve akış şemasını yazalım:

1. Oku Sayi1, Sayi2
2. Toplam=Sayi1+Sayi2
3. Yaz Toplam
4. Dur

9 ve 6 değerleri için bellekteki durum şöyledir:

Sayi1	Sayi2	Toplam
9	6	15



Örnek-2:

Klavyeden girilen 3 sayının aritmetik ortalamasını bulan programın algoritma ve akış şemasını yazın.

Değişkenler S1(1. sayı), S2, S3, Toplam, Ort (Ortalama) olmalıdır. S1, S2 ve S3 değişkenlerinin değerleri klavyeden okutulacaktır. Toplam ve Ort değişkenleri ise program içinde hesaplatılacaktır. Sonuç olarak ekranda Ort değişkeninin içeriği görüntülenecektir.

1. Oku S1, S2 ve S3

2. Toplam=S1+S2+S3

3. Ort=Toplam/3

4. Yaz "Ortalama=";Ort

5. Dur

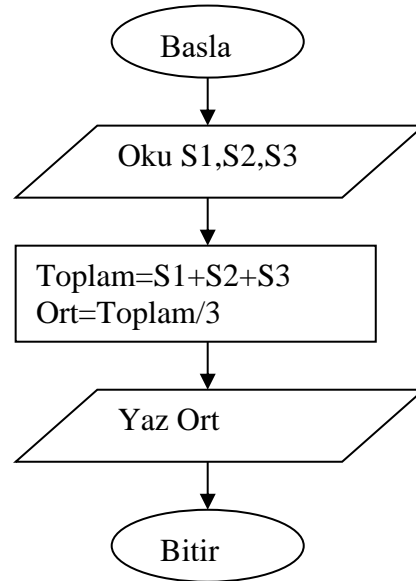
Yerine
Ort=(S1+S2+S3)/3
yazılabilir.

10,15,8 değerleri için bellekteki durum ve ekran çıktısı:

S1	S2	S3	Toplam	Ort
10	15	8	33	11

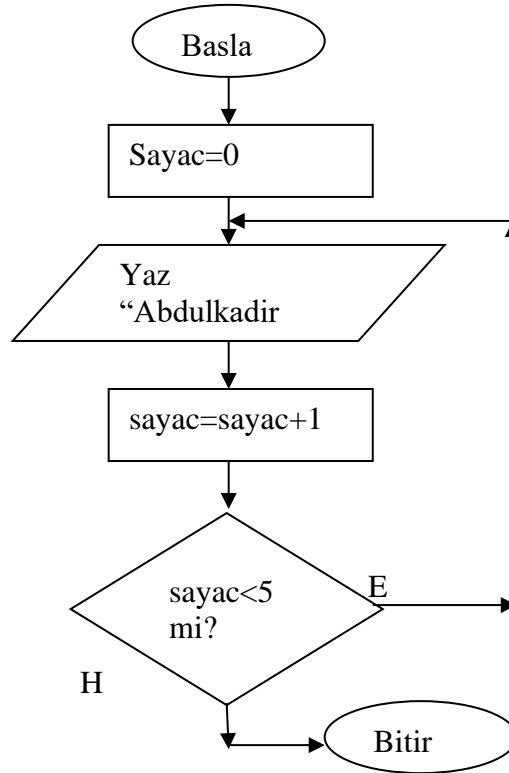
Ekran Çıktısı

Ortalama=11



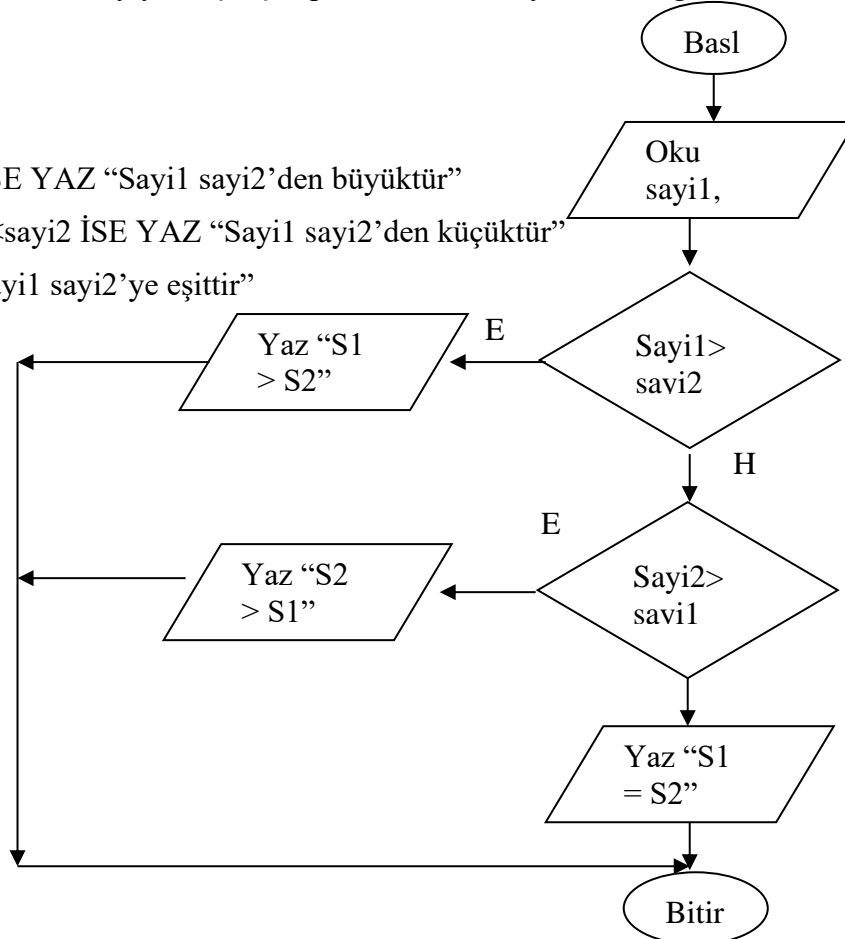
Örnek 3: İsim ve soyadınızı ekrana 5 defa yazdıran programın algoritma ve akış şemasını yazın?

1. Basla
2. sayac=0
3. YAZ “Abdulkadir KARACI”, sayac
4. sayac=sayac+1
5. Eğer sayac<5 GİT 3
6. DUR



Örnek 4: Klavyeden girilen 2 sayıyı karşılaştırıp sonucu ekrana yazdıran algoritma ve akış şemasını yazın?

1. BAŞLA
2. OKU sayi1,sayi2
3. EĞER sayi1>sayi2 İSE YAZ “Sayi1 sayi2’den büyüktür”
4. Değilse EĞER sayi1<sayi2 İSE YAZ “Sayi1 sayi2’den küçüktür”
5. DEĞİL İSE YAZ “Sayi1 sayi2’ye eşittir”
6. BİTİR

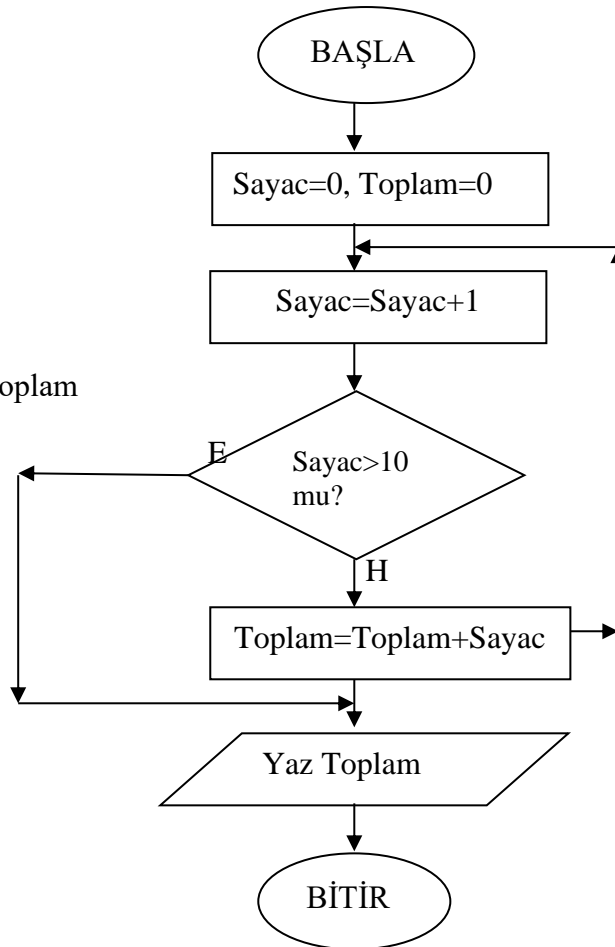


Ödev.1: Aşağıda verilen algoritmanın akış şemasını çizin ve programı izleyerek ne iş yaptığını belirtin?

1. BAŞLA
2. Oku sayi1, sayi2
3. Yaz Sayi1, Sayi2
4. Gecici=Sayi1
5. Sayi1=Sayi2
6. Sayi2=Gecici
7. Yaz Sayi1, Sayi2
8. Bitir.

Örnek 5 (inceleyin): 1-10 arasındaki tamsayıların toplamını bulan programın algoritma ve akış şemasını yazın?

- 1.BAŞLA
2. Sayac=0, Toplam=0
3. Sayac=Sayac+1
4. EĞER Sayac>10 İSE GİT 7
5. Toplam=Toplam+Sayac
6. GİT 3
7. YAZ “1-10 Arası Sayıların Toplamı=”,Toplam
8. BİTİR



Ödev 2: 1-10 arasındaki tamsayıların kareleri toplamını bulan programın algoritma ve akış şemasını yazın?

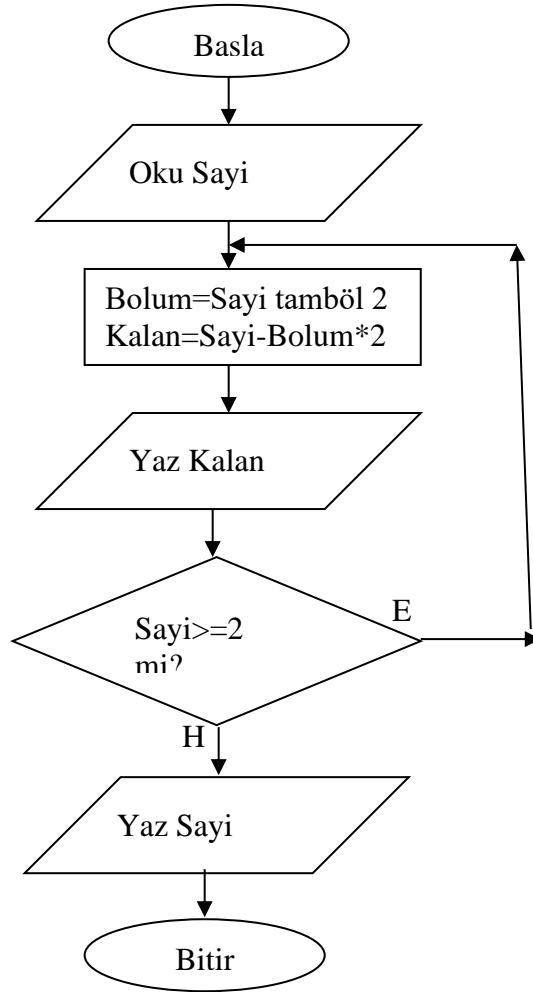
Örnek 6 (inceleyin): 1-100 arasındaki çift sayıların toplamını bulan programın algoritmasını yazın?

1. BAŞLA
2. Sayac=0, Toplam=0
3. Toplam=Toplam+Sayac
4. Sayac=Sayac+2
5. EĞER Sayac<=10 İSE GİT 3
6. YAZ “1-10 Arası Çift Sayıların Toplamı=”, Toplam
7. BİTİR

Ödev 3: 2-16 arasındaki çift sayıların ortalamasını bulan programın algoritma ve akış şemasını yazın?

Örnek 7 (inceleyin): Klavyeden girilen 10 tabanındaki sayıyı ikilik tabana çeviren programın algoritmasını ve akış şemasını yazın.

1. Basla
2. Oku Sayi
3. Bolum=Sayi tamböl 2
4. Kalan=Sayi-Bolum*2
5. Yaz kalan
6. Sayi=Bolum
7. Eger Sayi \geq 2 İse GİT 3
8. Yaz Sayi
9. Bitir



Örnek(inceleyin):

Sayi1=44.78

Sayi2=44.22

Ad="Mehmet"

Soyad=" Kara"

Deger1="123"

Deger2="456"

Yaz sayi1+sayi2

Yaz Ad+Soyad

Yaz Deger1+Deger2

Yandaki algoritmanın ekran çıktısını belirtin?

Ekran Çıktısı

```

89
Mehmet Kara
123456
  
```

Örnek(inceleyin):

A=6666

B=0

Yaz A/B

Programın Sonucu ne olur.

Bu program çalıştırıldığında 0'a bölme hatası ortaya çıkar. Bilgisayar Division by zero hatası verir.

Ödev

```

A=100
B=150
C=-100
D=A+B+C
Yaz A,B,C,D

```

```

X=50
Y=10+X
Z=X+X*Y
Yaz Z

```

```

isim="Kader"
A=1
A=A+1
Yaz A
Yaz isim

```

Soru: Yan taraftaki programların ekran çıktılarını tek, tek belirtin?

Ödev: A,B,C,D,E,F değişkenlerinin değerleri sırasıyla 10,5,12,6,3,20 ise aşağıdaki Yaz komutlarının ekran çıktısını yazın?

Yaz A,B,C

Yaz A;B;C

Yaz A*D,A+B+C,F/B,C+E

Yaz "B VE F'nin toplamı=";B+F

Örnek: Öyle bir algoritma yazın ki klavyeden ad ve soyad bilgisi girilsin. Daha sonra program ekrana Sayın (Girilen adsoyad) Programa hoş geldiniz yazısını yazsın? (Örneğin Adsoyad için Mehmet KARA girilmişse program Sayın Mehmet KARA Programa Hoş Geldiniz yazacak).

1. Basla
2. Oku "Ad Soyad Bilgisini Girin=";adsoyad
3. Yaz "Sayın"+adsoyad+"Programa Hoş Geldiniz"

Ad Soyad Bilgisini Girin=Mehmet KARA

Sayın Mehmet KARA Programa Hoş Geldiniz

ÖRNEK:

1. Başla
2. sayi1=5, sayi2=-20
3. Eğer sayi1<sayi2 ise
 gecici=sayi1
 sayi1=sayi2
 sayi2=gecici
4. Yaz "Sayı1=";sayi1
5. Yaz "sayi2=";sayi2
6. Bitir

Yukarıdaki algoritmayı izleyerek ekran çıktısını belirtin ve akış şemasını çizin.

ÖRNEK:

1. Başla
2. Oku notu
3. Eğer notu ≥ 50 ise yaz “GEÇTİ”
4. Bitir.

Eğer...İSE...DEĞİLSE Devimi**ÖRNEK:**

1. Başla
2. Oku notu
3. **Eğer** notu ≥ 50 **İse** yaz “GEÇTİ”
Değilse yaz “KALDI”
4. Bitir.

SORU: Klavyeden girilen 2 sayıyı karşılaştırıp; karşılaştırma sonucunu ekrana yazan algoritmayı yazın?

1. Başla
2. Oku “1. Sayıyı Girin...”;sayi1
3. Oku “2. Sayıyı Girin...”;sayi2
4. Eğer sayi1 > sayi2 İse Yaz sayi1 + ”>” + sayi2
5. Değilse Eğer sayi1 < sayi2 İse Yaz sayi1 + ”<” + sayi2
6. Değilse Yaz sayi1 + ”=” + sayi2

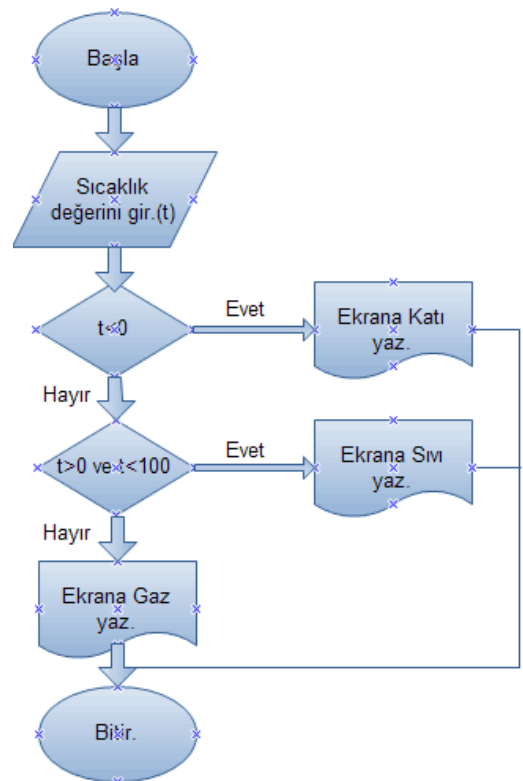
Örnek: Girilen sıcaklık değerine göre bir suyun katı, sıvı ve gaz olma durumunu gösteren programın algoritmasını ve akış şemasını tasarlayınız.

Şimdi soruyu çözmeden önce kimya konularını biraz hatırlayalım. Su sıfır derecenin altında katı, 0-100 derece arasında ise sıvı, 100 dereceden fazla ise gaz halinde bulunur. Dolayısı ile bu soruda kullanıcı su sıcaklığını girdikten sonra belirli karşılaştırmalar yaparak karar vermemiz lazım.

Değişkenler

Sıcaklık değeri: t

1. Başla
2. Oku “Sıcaklık değerini giriniz.”;t
3. Eğer $t < 0$ ise Yaz “Katı”.
4. Eğer $t > 0$ ve $t < 100$ ise Yaz ”sıvı” değilse yaz “gaz”
5. Bitir.



Ödev: Klavyeden girilen sayının negatif, pozitif ya da sıfır olduğunu ekrana yazan programın algoritmasını yazın.

Ödev: Bir dersten 3 sınav notu alan bir öğrencinin :

a- Ortalamasını

b- 5 li sistemdeki not karşılığını

c- Harfli sistemdeki not karşılığını yazdıran programın algoritmasını ve akış diyagramını tasarlayınız.

5 li sistemde verilen notlar:

100-85 dahil aralığı not 5 olur. Harfli sistemde A olur.

84-70 dahil aralığı not 4 olur. Harfli sistemde B olur.

69-55 dahil aralığı not 3 olur. Harfli sistemde C olur.

54-45 dahil aralığı not 2 olur. Harfli sistemde D olur.

44-25 dahil aralığı not 1 olur. Harfli sistemde E olur.

24-0 dahil aralığı not 0 olur. Harfli sistemde F olur.

Örnek: Bir fabrikada sabit maaşla çalışan işçiler aile durumlarına ve ürettikleri parça sayısına görede ek maaş almaktadır. Aşağıda verilen yönergelere göre işçilerin maaşlarını hesaplayan programın algoritmasını ve akış diyagramını tasarlayınız.

Çocuk sayısı 1 ise maaşın %5 i

Çocuk sayısı 2 ise maaşın %10 u

Çocuk sayısı 3 ve 3 den fazla ise maaşın %15 i kadar aile yardımı.

Üretilen parça sayısı 50-100 arasında ise maaşın %10

Üretilen parça sayısı 100-150 arasında ise maaşın %15

Üretilen parça sayısı 150-200 arasında ise maaşın %20

Değişkenler

İşçinin sabit maaşı:m

Çocuk sayısı:c

Ürettiği parça sayısı:p

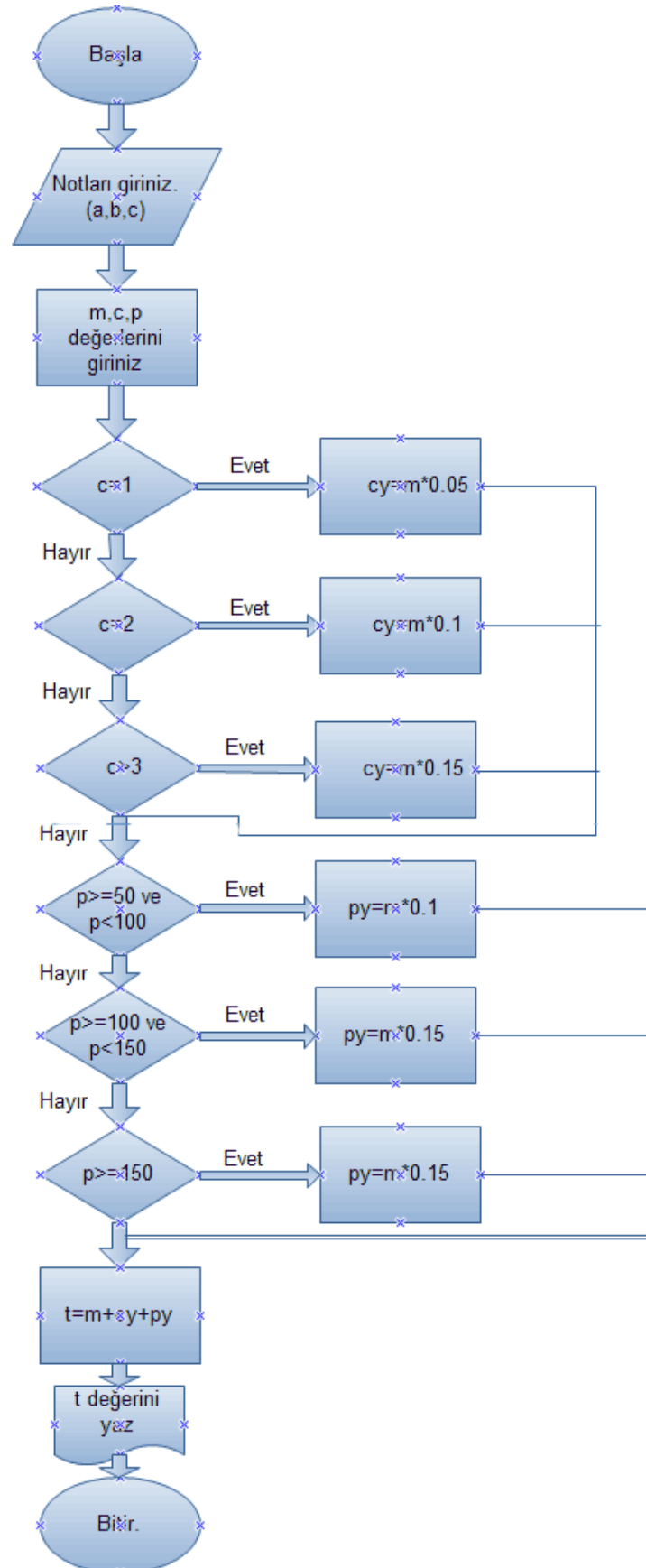
Çocuk yardımı: cy

Parça yardımı: py

Ödenecek toplam maaş: t

Algoritma

1. Başla
2. Oku "Sabit maaş",s
Oku "Çocuk sayısı",c
Oku "Üretilen Parça sayısı",p
3. Eğer c=1 ise $cy=m*0.05$
4. Eğer c=2 ise $cy=m*0.1$
5. Eğer c>2 ise $cy=m*0.15$
6. Eğer (p>=50 ve p<100) ise $py=m*0.1$
7. Eğer (p>=100 ve p<150) ise $py=m*0.15$
8. Eğer(p>=150) ise $py=m*0.2$
9. $t=m+cy+py$
10. t değerini ekrana yaz.
11. Bitir



Ödev:

Klavyeden 3 adet kenar uzunluğu giriliyor. Girilen kenar uzunlukları ile :

a-Üçgenin çizilip çizilemeyeceğini

b-Eğer üçgen çizilirse Üçgenin çeşidini(ikizkenar, çeşitkenar, eşkenar)

c-Çizilen üçgenin alan ve çevresini bulup ekrana yazan programın algoritmasını ve akış diyagramını tasarlayınız.

Yardım:

- Üçgenin üçgen olabilmesi için bir kenarı diğer iki kenarının toplamından küçük ve yine diğer iki kenarın farkından büyük olmak zorundadır. Bu özellik tüm kenarlar için doğru olursa belirtilen ölçülerde bir üçgen çizilebilir demektir.
- Üçgen tiplerinde eşkenar üçgen tüm kenarları eşittir, ikizkenar üçgen de herhangi iki kenar birbirine eşittir, çeşitkenar üçgende ise tüm kenar uzunlukları birbirinden farklıdır.
- Üçgenin çevresi tüm kenar uzunlukları toplamına eşittir. Alan formüllerinden biri ise $\text{Alan} = U \cdot ((U-a) \cdot (U-b) \cdot (U-c))^{1/2}$ U burada çevrenin yarısı kadardır. Yani yarı çevrede denilebilir. Bu bilgilerin ışığında algoritmayı yazın.

DÖNGÜ

Döngüler, aynı işlem birçok defa tekrarlandığında kullanılır. Örneğin ekrana yazma işleminin 100 defa tekrarlanması gereken bir durumda 100 tane “Yaz” komutu kullanmak yerine 1 tane yaz komutu kullanılır ve döngü kurulur. Döngü içinde işlem 100 defa tekrarlanabilir.

Soru: Klavyeden girilen Ad ve Soyad bilgisini ekrana 10 defa yazan algoritmayı yazın?

1. Başla
2. sayac=0
- 3.Okuy “Adınızı Ve Soyadınızı Girin=”;adsoyad
4. Yaz adsoyad
5. sayac=sayac+1
6. eğer sayac<3 ise Git 4
7. Bitir.

Bir çok programlama dilinde otomatik olarak döngü kurmak için özel komutlar vardır(For, While vb.). Biz algoritmada otomatik döngü olarak “Sayarak Tekrarla” ifadesini kullanacağız. Yukarıdaki algoritmanın otomatik döngü ile kodlanmış hali aşağıdaki gibidir.

1. Başla
2. Okuy “Adınızı Ve Soyadınızı Girin=”;adsoyad
- 3.Sayarak Tekrarla(sayac=1;sayac<=3;sayac++)
- 4.Yaz adsoyad
5. Tekrarlama Sonu
6. Bitir.

Soru: 1’den 100’e kadar olan sayılar içindeki çift sayıların toplamını bulan programın algoritmasını yazın?

1. Başla
2. Sayac=2,Toplam=0
3. Toplam=Toplam+sayac
4. Sayac=Sayac+2
5. Eğer sayac<=100 İse Git 3
6. Yaz “1-100 arası çift sayıların toplamı=”;toplam
7. Bitir

Ödev: “Sayarak Tekrarla” komutunu kullanarak yeniden kodlayın.

Soru: 1’den 100’e kadar olan sayılar içindeki tek sayıların toplamını bulan programın algoritmasını yazın?

Soru: Klavyeden girilen sayının tek sayı mı? Yoksa Çift sayı mı? Olduğunu bulup sonucu ekrana yazan programın algoritmasını yazın.

1. YOL

1. Başla
2. Oku “Bir Sayı Girin=”;sayi
3. Eğer $(-1)^{\text{sayi}}=1$ İse Yaz “Sayı Çifttir” Değilse Yaz “Sayı Tektir”
4. Bitir

2. YOL

1. Başla
2. Oku “Bir Sayı Girin=”;sayi
3. bolum=(sayi) tamböl (2)
4. Eğer bolum*2=sayi İse yaz “Sayı Çifttir” Değilse Yaz “Sayı Tektir”

Soru: Klavyeden girilen 10 sayının aritmetik ortalamasını ve toplamını bulup sonucu ekrana yazan programın akış şemasını ve algoritmasını yazın?

Örnek: Girilen n adet sayının:

- a-) 5 ile bölünebilen sayıların toplamı ve adeti
- b-) 3 ile bölünebilen sayıların toplamı ve adeti
- c-) 2 ile bölünebilen sayıların toplamı ve adeti

Değişkenler

Sayı adedi : n

Girilen sayı : s

Sayacımız: x

5 ile bölünebilenlerin sayısı : b

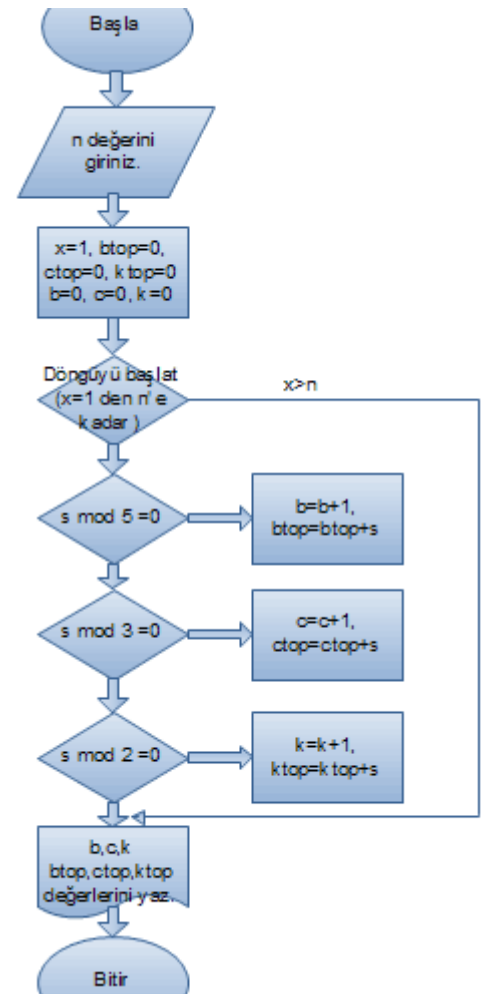
5 ile bölünebilen sayıların toplamı: btop

3 ile bölünebilenlerin sayısı : c

3 ile bölünebilen sayıların toplamı:ctop

2 ile bölünebilenlerin sayısı: k

2 ile bölünebilen sayıların toplamı : ktop



Algoritma

1. Başla
2. Oku “Girilecek sayı adedini giriniz”, n
3. $x=5$, $btop=0$, $ctop=0$, $ktop=0$, $b=0$, $c=0$, $k=0$
4. Sayarak Tekrarla($x=1, x \leq n, x=x+1$)
5. Oku $x+$ ”.Sayıyı Gir”,s
6. Eğer $s \bmod 5=0$ ise 5 e bölünebilir. $b=b+1$, $btop=btop+s$
7. Eğer $s \bmod 3=0$ ise 3 e bölünebilir. $c=c+1$, $ctop=ctop+s$
8. Eğer $s \bmod 2=0$ ise 2 e bölünebilir. $k=k+1$, $ktop=ktop+s$
9. Tekrarlama Sonu
10. Yaz b,c,k,btop,ctop,ktop
11. Bitir.

Ödev:

Klavyeden girilen şifre doğrultusunda(Şifre BLS), girilen bir sayının pozitif olup olmadığını kontrol eden ve sonuç pozitif ise girilen sayının faktöriyelini alan programın algoritmasını ve akış diyagramını bulunuz.

Şimdi bu örneğimiz de şifre yanlış girilirse program sonlandırılacak. Ayrıca girilen sayı 0 dan küçük ise yani negatif ise faktöriyel işlemi yapmıyacak. Bu arada bir sayının faktöriyeli demek sayının 1 değerine kadarki tüm değerlerinin birbiri ile çarpılması örneğin $4!=4*3*2*1$ gibi.

ALİŞTİRİM SORULARI

S.1. Aşağıda verilen programları izleyerek bellek değişkenlerinin değişimini ve ekran çıktısını belirtin?

1. Program

A= 10.430
B=-1.75
C=A*B
Yaz “A Değeri=”;A
Yaz “B Değeri=”;B
Yaz “C Değeri=”;C

2. Program

sayi=1
kare=sayi*sayi
sayi=sayi+1
kare=sayi*sayi
sayi=sayi+1
kare=sayi*sayi
Yaz “Sayı=”;sayi
Yaz “Kare=”;kare

3. Program

1. Başla
2. F=1
3. Sayi=5
4. F=F*Sayi
5. Sayi=Sayi-1
6. Eğer Sayi>1 İse Git 4
7. Yaz “Sonuç=”;F
8. Bitir.

4. Program

A=1250
B=0
Yaz A/B

5. Program

Sayac=0
Giris:
Oku “Sayı Girin”; Sayi
Toplam=Toplam+Sayi
Sayac=Sayac+1
Eğer Sayac<5 İse Git Giris
Ort=Toplam/Sayac
Yaz “Toplam=”;Toplam
Yaz “Ortalama=”;Ort

(Not: Girilen
Değerler:15,25,35,10,-10)

6. Program

Oku “Yaş Değerini Girin=”;Yas%
Eğer (Yas>=15) ve (Yas<=30) İse Yaz “Gençsiniz”
Eğer (Yas>30) ve (Yas<=45) İse Yaz “Orta Yaşlısınız”
Eğer (Yas>45) ve (Yas<=60) İse Yaz “Yaşlısınız”
Eğer Yas>60 İse Yaz “!!!!!!Ayağınız Çukurda!!!!!!”
(Not: Programı 15,42,70 ve 43 değerleri için izleyip ekran çıktısını belirtin.)

S.3. oku “Notunuzu Girin=”;Not
Eğer not>=50 İse Yaz “Başarılı”
Değilse Yaz “Başarısız”

S.4. Klavyeden girilen 3 sayının toplamlarını ve ortalamasını bulup sonucu ekrana yazdıran programın;
1.Algoritmasını yazın. 2.) Akış şemasını çizin. .

S.5. Aşağıda verilen algoritmanın akış şemasını çizin, programı izleyerek ne iş yaptığını belirtin?

1. BAŞLA
2. Sayi1=15
3. Sayi2=30
4. Yaz Sayi1, Sayi2
5. Gecici=Sayi1
6. Sayi1=Sayi2
7. Sayi2=Gecici
8. Yaz Sayi1, Sayi2

S. 6.

1. BAŞLA
2. Sayac=2, Toplam=0
3. Toplam=Toplam+Sayac
4. Sayac=Sayac+2
5. EĞER Sayac<=100 İSE GİT 3
6. YAZ “1-100 Arası Çift Sayıların Toplamı=”, Toplam
7. BİTİR

Yandaki algoritmada 4. satır kaldırılırsa programın çalışması ve çıktısı nasıl değişir.

S.7. Klavyeden girilecek iki sayıdan büyük olanından küçük olanını çıkarıp sonucu ekrana yazacak program için bir algoritmayı yazınız.

S.8. Girilen üç sayıdan en büyüğünü bulan algoritmayı yazınız.

S.9. Tamsayılarda üs alma işlemini gerçekleştiren algoritmayı yazınız (a^b).

S.10. 1-100 arasında tutulan bir sayıyı tahmin eden algoritmayı yazınız.

Örnek: Bir tamsayının faktöriyelini hesaplayınız.

Girdi: Bir tamsayı

Çıktı: sayının faktoriyel

İlgili formül: Faktoriyel(n) = $1*2*...*n$

1. n değerini oku
2. F=1
3. Sayarak Tekrarla(i=n; i > 1; i--)
3.1. F=F*i
4. Tekrarlama Sonu
5. F değerini yaz

Örnek : İki tamsayının bölme işlemini sadece çıkarma işlemi kullanarak gerçekleyin. Bölüm ve kalanın ne olduğu bulunacak.

1. a ve b değerlerini oku
2. $m=0$
3. Sayarak Tekrarla($i=a;i \geq b;i=i-b$)
 3.2 $m = m + 1$
4. Tekrarlama Sonu
5. kalan i ve bölüm m 'yi yaz

Örnek: 100 tane sayıyı okuyup, ortalamasını bulun.

1. $T=0, i=0$
2. Sayarak Tekrarla($i=1;i \leq 3;i++$)
 2.1 m değerini oku
 2.2 $T = T + m$
3. Tekrarlama Sonu
4. $T = T / 3$
5. Ortalama T 'yi yaz
6. Dur

Çalışma Soruları:

- S.1. Klavyeden girilen isim değerini klavyeden girilen sayı kadar ekrana yazdıran algoritmayı yazın?
- S.2. Klavyeden girilen 2 değer arasındaki çift sayıların toplamını hesaplayan algoritmayı yazın.
- S.3. Klavyeden girilen sayının asal sayı olup olmadığını yazan algoritmayı yazın?
- S.4. Klavyeden girilen 2 değer arasındaki 5'e bölünebilen sayıların toplamını bulan algoritmayı yazın?
- S.5. Klavyeden girilen sayı kadar, klavyeden girilen sayıların ortalaması bulup ekrana yazdıran programın algoritmasını yazın?
- S.6. Dışarıdan girilen rastgele 10 tane sayıdan tek ve çift sayıların ortalamasını ayrı, ayrı bulup ekrana yazan programın algoritmasını yazın?
- S.7. Dışardan iki sayı okuyup 1. sayıyı taban 2. sayıyı üs kabul ederek üs alma işlemini yapan programın algoritmasını yazınız.
- S.8. Dışardan okunan 10 tane rast gele sayıdan kaçının negatif kaçının pozitif olduğunu ve pozitifleri kendi arasında negatifleri kendi arasında toplayıp sonuçları ekrana yazan programın algoritmasını yazınız.
- S.9. Sadece toplama işlemi kullanarak girilen iki sayıyı çarpan programın algoritmasını yazınız

C ve C++ PROGRAMLAMA DİLİ VE PROGRAM ÖRNEKLERİ

Bir C programı bir veya daha fazla fonksiyondan oluşmakta ve her bir fonksiyon bir veya daha fazla sayıda deyim içermektedir. C'de bir fonksiyon altıyordam (*subroutine*) olarak da adlandırılır ve bunlar programın bir başka yerinden isimleri kullanılarak çağrılabilir.

```
#include<stdio.h>

main()
{
    printf("Türkiyem");
}
```

Program : Belirli bir problemi çözmek için bir bilgisayar dili kullanılarak yazılmış deyimler dizisi. Önceki bölümde bir problemin çözümü ile ilgili teknikler sunmuştuk. Bir problemi bilgisayar ile çözmek için geliştireceğimiz programın yazımında izleyeceğimiz adımlar:

- Problemin ne olduğunu kavra. Çözüm için gereksinimleri belirle.
- Problemin girdilerini, çıktılarını ve diğer kısıtlama ve gereksinimleri belirle (bilgilerin giriş ve çıkış biçimlerinin nasıl olacağına kadar).
- Problemin çözümünü veren algoritmayı yaz.
- Algoritmayı bir programla dili ile yaz.
- Programın doğru çalışıp çalışmadığını test et. Bu testi değişik veriler (girdiler) için tekrarla.

1. Programı Derleme / Yorumlama

Bilgisayarların çalıştıracakları programların belli bir biçimi olması zorunludur. Bu biçim, bilgisayardan bilgisayara ve işletim sisteminden işletim sistemine göre farklılıklar gösterir. Sözelimi, bir kişisel bilgisayar üzerinde Windows işletim sisteminde çalışan bir program, Sun marka bir işstasyonunda Solaris işletim sisteminde çalışmaz. Hatta, yine aynı kişisel bilgisayar üzerinde Linux işletim sisteminde de çalışmaz. Programların bu çalıştırılabilir biçimine ``*makina kodu*" (machine code) adı verilir. Makina kodu, insanların anlaması ve üzerinde çalışması son derece zor bir biçim olduğundan programcılar programları doğrudan bu kod ile yazmazlar. İnsanın anlaması daha kolay (insan diline daha yakın) ``yüksek düzeyli" (high-level) bir dil ile yazıp yardımcı yazılımlar aracılığıyla makina koduna çevirir ve çalıştırlar. Programcının yazdığı bu koda ``*kaynak kodu*" (source code) denir.

Kaynak kodunun makina koduna çevrilmesi ve çalıştırılması işlemi üç farklı yaklaşımla gerçekleştirilebilir:

Yorumlama

Programın komutları bir yorumlayıcı (interpreter) tarafından *teker teker* okunur, makina koduna çevrilir ve çalıştırılır. Yani yorumlayıcı, önce birinci komutu okur, makina koduna çevirir ve çalıştırır. Sonra ikinci komutu alır ve aynı işlemleri yapar. Programın her çalışmasında çevirme işlemi yeniden yapılır. Herhangi bir komutta bir hatayla karşılaştığında bunu kullanıcıya bildirir ve çalışmayı durdurur. Basic, Perl, Python gibi diller genellikle yorumlayıcılar aracılığıyla kullanılırlar.

Derleme

Program bir derleyici (compiler) tarafından *bir bütün halinde* okunur, makina koduna çevrilir ve çalıştırılır. Çevirme işlemi sonucunda bir çalıştırılabilir dosya oluşur. Çevrilmiş olan program daha sonra çalıştırılır, yani çevirme işlemi yalnızca bir kere yapılır. Herhangi bir komutta bir hata varsa

çevirme işlemi tamamlanmaz ve çalıştırılabilir kod oluşturulmaz. Fortran, Pascal, C gibi diller genelde derleyicilerle kullanılmakla birlikte bunları işleyen yorumlayıcılar da bulunabilmektedir.

Karma

Hem derleme hem de yorumlama tekniği kullanılır. Bu tip çalışmada kaynak kodu sanal bir bilgisayarın makina koduna (bytecode) çevrilir ve daha sonra bu sanal bilgisayarı gerçekleyen bir program yardımıyla yorumlanarak çalıştırılır. Örneğin Java dilinde yazılmış bir kaynak kodu önce Java derleyicisinden geçirilerek Java sanal makinasının (Java Virtual Machine - JVM) makina koduna dönüştürülür; sonra da bu sanal makinaı gerçekleyen bir Java çalışma ortamı (Java Runtime Environment - JRE) yardımıyla çalıştırılır.

Yorumlama yönteminde kodun okunması ve çevrilmesi programın çalışması sırasında yapıldığından hız düşüktür. Ayrıca yorumlayıcı yalnızca karşılaştığı ilk hatayı rapor edebilir. Bu hata düzeltildiğinde sonraki çalışmada da program ancak bir sonraki hataya kadar ilerleyebilir. Oysa derleyici kaynak kodundaki bütün hataları bulabilir.

Buna karşılık hata ayıklama -özellikle deneyimsiz programcılar için- yorumlayıcılarla daha kolaydır. Derleyicilerle gelen bazı sınırlamaların kalkması nedeniyle daha esnek bir çalışma ortamı sağlanır.

2. Kitaplıklar

Bir programcıya gerekebilecek her şeyi programlama dilinin içine almak o dil için yazılan derleyicilerin hantallaşmalarına neden olur. Örneğin basit aritmetik işlemlerin ötesindeki matematik işlemleri dilin tanımı içinde yer almaz.

Bununla birlikte, pek çok programcının bu tip işlemlere gereksinim duyacakları da açıktır. Böyle bir durumda programcı, isterse karekök alma işlemini yapacak kodu kendisi yazabilir. Ancak her programcının kendine gereken işlemler için kendi kodlarını yazmasının önemli sakıncaları vardır:

1. Programcının yazacağı kod hatalı olabilir. Karekök alma işlemi yapan kod yanlış sonuç üretebilir.
2. Programcının yazacağı kod yeterince etkin olmayabilir, yani işlemi yapmak için gereğinden fazla zaman ya da sistem kaynağı harcayabilir.
3. Çok sayıda programcının aynı işleri yapan kodlar yazmaları büyük bir zaman yitirilmesine yol açar.

Hem dilin tanımını küçük tutmak hem de bu sakıncaları giderebilmek için, çok sayıda programcıya gerekebilecek işlemler (yordamlar) *kitaplık* (library) adı verilen arşivlerde toplanmıştır. Bir sayının karekökünü almak isteyen bir programcı matematik kitaplığındaki `sqrt` yordamını kullanabilir. Kitaplığın kullanılması yukarıda sözü geçen sakıncaları giderir, yani programcıya zaman kazandırdığı gibi, doğru ve etkin çalıştığı sınanmış olduğundan dikkatini programın diğer kısımlarına yoğunlaştırma fırsatı verir.

Örnek: `stdio.h`, `conio.h`, `math.h`, `ctype.h`

- **Standard Input/Output (stdio) kitaplığı**
 - **#include <stdio.h>**
- Tek karakter **okuma**
 - **stdin** den tek karakter oku
 - **c = getchar();**
- Tek karakter **bas**
 - **stdout** a tek karakter bas
 - **putchar(c);**

İlk C Programı Örnekleri

	C
Program	hello.c: #include <stdio.h> int main(void) { printf("Hello, world\n"); return 0; }
Derleme (Compile)	% gcc hello.c % ls a.out hello.c %
Çalıştırma (Run)	% a.out Hello, world %

```
#include <stdio.h>           //Kullanılan işlevler ile ilgili kütüphane dosyası
main()
{
    int i ;                  /*Değişken tanımlama*/
    scanf("%d",&i);          //sayı okunuyor
    i=i*i;                   //sayının karesi alınıyor
    printf("%d",i);          //sonuç yazdırılıyor
}
```

Main() Fonksiyonu

Bir C programı bir veya daha fazla fonksiyondan oluşmaktadır. Önceki sayfalarda verdiğimiz örneklerde yer alan main() bir fonksiyondur. Bir C programının içinde çok sayıda fonksiyon yer alabilir, ancak mutlaka bir main() fonksiyonu bulunmalıdır.

Program yürütülmeye main() fonksiyonundan başlar. C'de programcı tarafından tanımlanan ve isimlendirilen fonksiyonlar kullanılabilir. Bu tür fonksiyonların nasıl oluşturulabileceği konusunu dersimizin ilerideki bölümlerinde ele alarak inceleyeceğiz. C' de programcı tarafından tanımlanan fonksiyonlar dışında, hazır kitaplık fonksiyonları da bulunmaktadır.

Örneğin, ekran üzerine mesaj yazdırmak için kullanılan printf(), bir kitaplık fonksiyonudur.

3. C Veri Tipleri

<u>Veri Tipi</u>	<u>Açıklama</u>	<u>Bellekte Kıstırdığı Alan (byte)</u>	<u>Alt Sınır</u>	<u>Üst Sınır</u>
char	Tek bir karakter veya küçük tamsayı için	1 (8 Bit)	-128	127

unsigned char	İşaretsiz char		0	255
short int	Kısa tamsayı için	2 (16 Bit)	-32,768	32,767
unsigned short int	İşaretsiz kısa tamsayı		0	65,535
int	Tamsayı için	4 (32 Bit)	-2,147,483,648	2,147,483,647
unsigned int	İşaretsiz tamsayı		0	4,294,967,295
long int	Uzun tamsayı için	8 (64 Bit)	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long int	İşaretsiz Uzun tamsayı		0	18,446,744,073,709,551,615
float	Tek duyarlı gerçel sayı için (7 basamak)	4 (32 Bit)	-3.4e +/- 38	+3.4e +/- 38
double	Çift duyarlı gerçel sayı için (15 basamak)	8 (64 Bit)	-1.7e +/- 308	+1.7e +/- 308

veri_türü değişken_adı;

int sayac;

int sayac=0

float katsayi=3.14

Soru: İşaretsiz tamsayı değişkenlerin erimleri neden işaretli tamsayılardan daha yüksektir.

Giriş-Çıkış Devimleri (C ve C++)

- **printf ve scanf Deyimleri**

printf (“kontrol karakterleri”, argüman listesi)

% İşareti İle Kullanılan Kontrol Karakterleri	
d veya i	decimal (integer)
u	unsigned decimal
c	char (tek karakter)
s	String (karakter dizisi)
e	float/double sayıyı bilimsel gösterimde yaz

f	float/double sayıyı [-] mmm.nnnnn biçiminde yaz
ld	long integer
lu	unsigned long integer
le,lf	long double
lx	Uzun tamsayılar onaltılık (Hex) sistemde
x	Tamsayılar için (Onaltılık (Hex) sistemde küçük harfle)
X	Tamsayılar için (Onaltılık (Hex) sistemde büyük harfle)
o	Tamsayı değişkenler (Sekizlik sistemde)

\ İşareti ile Başlayan Kontroller

Karakter	İşlev tanımı
\n	Yeni satıra geçme
\t	Yatay olarak bir sekme atlama
\r	Bulunulan satırın başına geçme
\a	Alarm (Beep sesi)
\\	Ters bölme işareti
\'	Tek tırnak
\"	Çift tırnak

Ödev Soru:

Problem: Dosya içeriğini büyük harfe çevirecek bir program yazın.

Program Tasarımı:

tekrarla

bir karakter oku

küçük harfse, büyük harfe çevir

karakteri yaz

dosya sonuna ulaşıncaya kadar

İpucu: EOF kavramı ve ASCII tablosu araştırması yapabilirsiniz.

Örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char k='A';
    float s1=123;
    double s2=607.591;
    printf("Karakter:%c\n",k);
    printf("Sayi1:%f\n",s1);
    printf("Sayi2:%f\n",s2);
}
```

```
Karakter:A
Sayi1:123.000000
Sayi2:607.591000
Sayi2:607.59
Sayi2:607.6
Sayi2:608
```

```

printf("Sayi2:%3.2f\n",s2);
printf("Sayi2:%3.1f\n",s2);
printf("Sayi2:%3.0f\n",s2);
getch();
}

```

- **scanf (kontrol karakterleri, argüman listesi)**

```

int a,b,c;
float m,n;
scanf("%d", &a);           //Klavyeden tamsayı okur. Girilen değer a
                           //değişkenine aktarılır.
scanf("%d %d",&a,&b)       //Klavyeden girilen ilk değer a değişkenine,
                           // ikinci değer b değişkenine aktarılır.
scanf("%f %d", &m, &a);    // Klavyeden ilki gerçel, ikincisi tamsayı olmak
                           //üzere iki değer okur.

```

- **cout deyimi**

- C++’da standart çıkış nesnesine yönlendirme yapmak için “<<” işleci kullanılır.
- Ekran nesnesine karşılık gelmek üzere cout sözcüğü kullanılır.

```

cout << "Merhaba Turkiye"<<endl;
int a=5;
cout<<"a="<<a<<"\n";

```

- **cin deyimi**

- Eğer bir klavyeden veri girişi söz konusu ise, “>>” işleci kullanılır.
- Klavye nesnesi için cin sözcüğü kullanılır.
- Söz konusu bilgi giriş çıkış işlemlerini yapabilmek için, ilgili kitaplıkları içeren iostream dosyasının programa dahil edilmesi gerekmektedir.
- #include <iostream >

```

int sayi;
cin >> sayi;

```

Örnek:

```

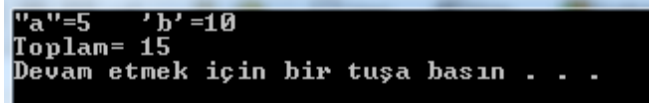
#include <iostream>
using namespace std;

```

```

int main()
{ int a=5,b=10;
  cout<<"\a\"<<a
    <<"\t\b\"<<b<<"\n";
  cout<<"\t"<<a+b<<"\rToplam="<<"\n";
  system("PAUSE");
  return 0;
}

```



```

'a'=5 'b'=10
Toplam= 15
Devam etmek için bir tuşa basın . . .

```

Değişkenler

C programları içinde farklı amaçlara yönelik değişken tanımları yapılabilir. Değişken türlerini aşağıdaki şekilde sıralayabiliriz:

- Yerel Değişkenler
- Küresel Değişkenler
- Extern Değişkenler
- Static Değişkenler
- Auto Değişkenler
- Register Değişkenler

```
Deger1=5
Deger2=5
```

Yerel Değişkenler:

- Değişken veri türü bildirimleri bir fonksiyonun içinde ya da dışında yapılabilir.
- Fonksiyon içinde bildirimi yapılan bir değişken, sadece o fonksiyon için geçerlidir.
- Yerel değişkenler, program yürütüldüğünde aktif hale geçerek kendisi için ayrılan bellek alanlarını kullanır.
- Ancak yer aldıkları blok sona erdiğinde bu bellek alanları iptal olur ve değişken içeriği tamamen yok olur.
- Aynı blok daha sonra tekrar başlasa bile, yerel değişkenler eski değerlerini alamaz.

<pre>#include<stdio.h> #include<conio.h> void fonksiyon1(); void fonksiyon2(); int i=5; //Global değişken main() { int i=10; //Yerel Değişken printf("Değer1=%d\n",i); fonksiyon1(); printf("Hangisini Yazar=%d\n",i); fonksiyon2(); getch();} </pre>	<pre>void fonksiyon1() { int i=15; printf("Değer2=%d\n",i); } void fonksiyon2() { printf("Hangisini Yazar=%d\n",i); } </pre>	<pre>Deger1=10 Deger2=15 Hangisini Yazar=10 Hangisini Yazar=5</pre>
--	---	---

Global Değişkenler:

Eğer bir değişkenin program içindeki tüm fonksiyonlar için geçerli olması isteniyorsa, fonksiyonların dışında bir yerde tanımlanmalıdır.

```
#include<stdio.h>
#include<conio.h>
void fonksiyon1();
int i=5;
main()
{
    printf("Deger1=%d\n",i);
    fonksiyon1();
    getch();
}

void fonksiyon1()
{
    printf("Deger2=%d\n",i);
}
```


Extern Değişkenler:

- Global değişkenlerin içerdiği değerlerin, programın sonuna kadar tüm fonksiyonları kapsayacak biçimde geçerli olduğunu biliyoruz.
- Bazı durumlarda bir program dosyası içinde tanımlanan küresel değişkenlerin, bir başka programda da geçerli olması istenilebilir. Böyle durumlarda **extern değişkenler** kullanılır.

<u>Header.h</u>	<u>Ana Program</u>
<pre>#include <iostream> using namespace std; extern int externdegisken=10; extern void ekranayaz(int i) { cout<<"Extern Degisken="<<i<<endl; }</pre>	<pre>#include <iostream> #include "header.h" using namespace std; int main() { externdegisken=15; ekranayaz(externdegisken); system("PAUSE"); return 0; }</pre>

```
Extern Degisken=15
Devam etmek için bir tuşa basın . . .
```

Statik Değişkenler:

- Normal olarak, içinde lokal değişken tanımlanan bir fonksiyonu her çağırmanızda, lokal değişken değeri ilk atanan değer olur.
- Ancak, bir fonksiyon içinde static lokal bir değişken tanımladığınızda, fonksiyonu her çağırmanızda lokal değişken bir önceki fonksiyon çağırısındaki en son değerini korur.
- Değişkendeki bir önceki değer kaybolmaz.

```
#include <stdio.h>
#include <conio.h>
void fonk1 (void);
void fonk2 (void);
main()
{
    fonk1();
    fonk2();
    printf("\n");

    fonk1();
    fonk2();
    printf("\n");

    fonk1();
    fonk2();
    getch();}

void fonk1 (void)
{
    int id1 = 1;
    printf("%d ", id1);
    id1 = id1 + 5;
    printf("%d ", id1);
}

void fonk2 (void)
{
    static int id1 = 0;
    printf("%d ", id1);
    id1 = id1 + 9;
    printf("%d ", id1);
}
```

```
1 6 0 9
1 6 9 18
1 6 18 27
```

Register Değişkenler:

- Önceki sayfalarda incelediğimiz değişkenlerin tümü bellek üzerindeki alanları kullanır.
- Bazı uygulamalarda, değişkenlere çok daha hızlı biçimde erişmek söz konusu olabilir.
- Böyle durumlarda, register değişkenleri tanımlanarak bellek alanları yerine, makinenin hızlı bellek yazmaçları kullanılabilir.
- Ancak bu alanların miktarı sınırlı olduğundan, fazla sayıda register değişken tanımlanırsa amacına ulaşmaz. Derleyici fazla bulduklarını normal bellek alanlarına yerleştirir.

```
#include <iostream>
using namespace std;
int main()
{ register int i=100;
  i=i+100;
  cout<<i<<endl;
  system("PAUSE");
  return 0;
}
```

```
#include<stdio.h>
main()
{ register int k;
  { register int i;
    ...
  }
}
```

Değişkenlere İsim Verilirken Dikkat Edilecek Kurallar

- Her programlama dilinin kendine ayırdığı ve komutları için seçtiği isimler vardır. C dili içerisinde anahtar kelimeler, komut veya fonksiyon adları değişken ismi olarak kullanılamaz.
- Değişken isimleri içerisinde a-z ve A-Z arası İngiliz harfleri, 0-9 arası rakamlar ve özel karakter olarak yalnızca “_” (alt çizgi) kullanılabilir. Özel karakterler (+,-,! vs.) ve Türkçe karakterler (ÜüÇçĞğİıÖöŞş) kullanılamaz. Örneğin maaş, öğrenci, sınıf değişken ismi olamazlar.
- Değişken ismi rakamla başlayamaz, fakat daha sonra rakam kullanılabilir. **1.vize** adında değişken olamaz ama **vize1** olabilir.
- İki kelimeden oluşan değişken isimleri arasına ya alt çizgi (_) ya da ikinci kelimenin baş harfi büyük yazılarak kolaylaştırılmalıdır.
- Değişken isimleri istenilen uzunlukta olabilir.

Sabitler

Veri türü const sabit_adı = değeri;

Bir sabit tanımlandıktan sonra, programın herhangi bir yerinde değerini değiştirmek olanaksızdır.

```
#include <iostream>
using namespace std;
float const PI=3.14;
char const cinsiyet='E';
int main()
{ cout<<"PI="<<PI<<"\n";
  cout<<"Cinsiyet="<<cinsiyet<<"\n";
  system("PAUSE");
  return 0;}
```

```
PI=3.14
Cinsiyet=E
Devam etmek için bir tuşa basın . . .
```

Matematiksel İşlemler

Matematikteki	C dilindeki	İşlem
.x	*	Çarpma
/	/	Bölme
+	+	Toplama
-	-	Çıkarma
Mod	%	Modüler bölüm
-1	--	1 Eksiltme
+1	++	1 Artırma
a ²	^	Üs alma

A++	Önce A'yı kullan, sonra içindeki değeri 1 artır
++A	Önce A'yı 1 artır, sonra bu artırılmış değeri kullan
A--	Önce A'yı kullan, sonra içindeki değeri 1 eksilt
--A	Önce A'yı 1 eksilt, sonra bu eksiltilmiş değeri kullan

a=4 ve b=5 olsun.

İşlem	İşlemlerin gerçekleştirilme sırası		Değişkenlerin son değerleri		
c=a++ *b	c=a*b a=a+1	c=4*5=20 a=4+1=5	a=5	b=5	c=20
c=++a *b	a=a+1 c=a*b	a=4+1=5 c=5*5=25	a=5	b=5	c=25
c=++a -b	a=a+1 c=a-b	a=4+1=5 c=5-5=0	a=5	b=5	c=0
c=--a + b--	a=a-1 c=a+b b=b-1	a=4-1=3 c=3+5=8 b=5-1=4	a=3	b=4	c=8

Mantıksal İşlemler

>	Büyük
<	Küçük
<=	Küçük Eşit
>=	Büyük Eşit
==	Karşılaştırma
!=	Eşit Değil
&&	Ve (And)
	Veya (Or)
&	Ve (And)
	Veya (Or)
!	Değil (not)

Araştırma Sorusu: && ve & işleci arasında ne fark vardır. Benzer araştırmayı || ve | işlemleri için de yapın.

Atama İşleçleri

=	Atama
+=	Toplayarak atama
-=	Eksilterek atama
*=	Çarparak atama
/=	Bölerek atama
%=	Mod alarak atama

Örnek: Aşağıdaki programı x=10, y=2 için izleyerek ekran çıktısını ve bellek değişkenlerinin değişimini gösterin. (Aşağıdaki örnek C++'da yazılmıştır)

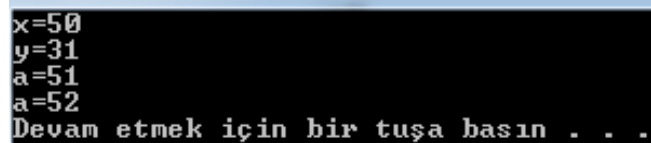
```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    float x,y;
    int z;
    cout<<"x : ";
    cin>>x;
    cout<<"y : ";
    cin>>y;
    cout<<".....Sonuçlar yazdırılıyor....."<<endl;
    x+=y;
    cout<<"x = "<<x<<endl;
    x-=y;
    cout<<"x = "<<x<<endl;
    x*=y;
    cout<<"x = "<<x<<endl;
    x/=y;
    cout<<"x = "<<x<<endl;
    z=8;
    z%=5;
    cout<<"z % 5 = "<<z<<endl;}

```

Örnek:

```
#include <iostream>
using namespace std;
int main()
{
    int a=50;int b=30;
    int x,y;
    x=a++;
    y=++b;
    cout<<"x="<<x<<"\n";
    cout<<"y="<<y<<"\n";
    cout<<"a="<<a++<<"\n";
    cout<<"a="<<a<<"\n";
    system("PAUSE");
    return 0;
}

```



```
x=50
y=31
a=51
a=52
Devam etmek için bir tuşa basın . . .

```

Örnek:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 2,b=-1,c=2,d=5;
    b +=-- ++a;
    ++c +=a++;
    d/=c;
    d*=a++;
    cout<<"a="<<a++<<"\t"<<"b="<<++b<<"\t"
    <<"c="<<c<<"\t"<<"d="<<d<<"\n";
    system("PAUSE");
    return 0;
}
```

```
a=4      b=2      c=5      d=3
Devam etmek için bir tuşa basın . . .
```

Örnek :

```
#include <iostream>
using namespace std;
int main()
{
    int s1=10;int s2=1;
    s1=s1--;
    cout<<s1<<"\n";//10 yazar
    system("PAUSE");
    return 0;
}
```

10 yazar çünkü işlem soldan sağa doğru yapılır.
S1 değişkeninin değeri önce temp alana alınır.
s1 bir eksiltirilir
Sonra temp alandaki değer s1'e aktarılır.
S1:10,9,10 şeklinde değişir.

```
10
Press any key to continue . . .
```

Örnek :

```
#include <iostream>
using namespace std;
int main()
{
    int s1=10;int s2=1;
    s1=s1+s2+s1--;
    cout<<s1<<"\n";//21 yazar
    system("PAUSE");
    return 0;
}
```

21 yazar çünkü işlem soldan sağa doğru yapılır.
s1 ve s2 toplanır (10+1=11).
s1'in değeri temp alana atılır sonra bir eksiltirilir.
Sonra temp alandaki değer diğer değerlerle toplanır (11+10=21).
S1:10,9,21 şeklinde değişir.

```
21
Press any key to continue . . .
```

Örnek :

```
#include <iostream>
using namespace std;
int main()
{
    int s1=10;int s2=1;
    s1=s1--s2+s1;
    cout<<s1<<"\n";//20 yazar
    system("PAUSE");
    return 0;
}
```

20 yazar çünkü işlem soldan sağa doğru yapılır.

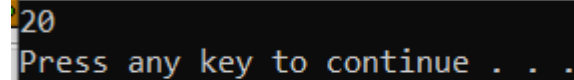
s1 temp alana atılır sonra bir eksiltir..

Sonra temp alandaki değer ile s2 toplanır

(10+1=11).

Sonra s1'in eksiltilmiş değeri 9 ile diğer değerler toplanır (11+9=20).

S1:10,9,20 şeklinde değişir.



Örnek : Yarıçapı belli dairenin alanını hesaplayan programı yazınız.

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    float r,alan;
    cout<<"Yarıçapı Girin : ";
    cin>>r;
    alan=3.14*r*r;
    cout<<"Alanı : "<<alan<<endl;
}
```

Siz Çözün: Klavyeden yarı çapı girilen dairenin alanını ve çevresini hesaplayan C programını yazın.

Örnek: $ax^2+bx+c=0$ tipi bir denklemin köklerini veren programı yazınız.

Girdi : a, b ve c katsayıları

Çıktı : denklemin kökleri

Algoritma :

1. a, b ve c katsayılarını oku.
2. Delta= değerini hesapla.
3. x_1 ve x_2 değerlerini hesapla.
4. Kökleri yaz.

Programın kodlanması:

```

#include <iostream>
#include<cmath>
using namespace std;

int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter

    float a, b, c;
    double x1, x2;
    double d;
    cout<<"a : ";    //1 giriniz
    cin>>a;
    cout<<"b : ";    //-3 giriniz
    cin>>b;
    cout<<"c : ";    //2 giriniz
    cin>>c;
    d = b * b - 4 * a * c;
    x1 = (-b + sqrt(d)) / (2 * a);
    x2 = (-b - sqrt(d)) / (2 * a);
    cout<<"x1'in değeri : "<<x1<<endl<<"x2'nin değeri : "<<x2<<endl;
}

```

C dilinde tanımlı bazı istatistiksel fonksiyonlar aşağıda listelenmektedir. Bu fonksiyonların hepsi “**cmath**” veya “**math.h**” sınıfı içinde tanımlıdır.

<i>Fonksiyon</i>	<i>x ,y</i>	<i>Sonuç</i>	<i>Gerçekleştirilen İşlem</i>
Abs(x)	decimal	decimal	x'in mutlak değeri
Pow(x, y)	double	double	x^y
Sqrt(x)	double	double	x'in karekökü
Exp(x)	double	double	e^x değeri
Log(x)	double	double	$\ln(x)$ değeri
Log10(x)	double	double	$\log_{10}(x)$ değeri
Ceiling(x)	double	int	x ten büyük ilk tamsayı
Floor(x)	double	int	x ten küçük ilk tamsayı
Round	double	int	Yuvarlama ya da kesme yapar

Örnekler:

ceiling(5)	Hata
ceiling (5.2)	6
ceiling (-5.2)	-5
floor(5)	Hata
floor(5.2)	5
floor(-5.2)	-6

C'de Rastgele Sayı Üretme

C'de rastgele sayı üretmek için rand() komutu kullanılmaktadır. Rand() işlevi, C/C++'da [0, RAND_MAX) aralığında rasgele sayılar üretmek için kullanılır. int Rand(void): [0, RAND_MAX) aralığında bir rasgele sayı döndürür.

RAND_MAX: varsayılan değeri uygulamalar arasında değişebilen ancak en az 32767 olarak verilen bir sabittir.

ÖLÇEKLEME

Belirli aralıkta sayı üretmek için aşağıdaki ölçekleme kullanılır.

rand() % a + b

a : ölçekleme faktörü

b : kaydırma değeri

Örnek:

```
rand() % 6      // [0,5] aralığında sayı üretir.
rand() % 6 + 1  // [0+1,5+1] = [1,6] aralığında sayı üretir.
```

Soru: [0-1] aralığında rasgele sayı nasıl üretirsiniz?**Örnek:**

```
#include <stdio.h>
#include <stdlib.h> //rastgele sayı için bu kütüphane dahil edilmelidir.
int main(void)
{
    for(int i = 0; i<5; i++)
        printf(" %d ", rand()%50); //[0-49] aralığında 5 sayı üretir.
    return 0;
}
```


Örnek: Klavyeden girilen vize ve final notlarından ortalamayı hesaplayan C programını yazın.

```
#include <iostream>
using namespace std;

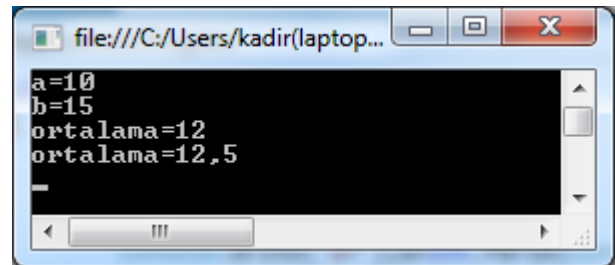
int main()
{
    float v_puan, f_puan, ort;
    cout<<"Sinav vize puanini giriniz : "<<endl;
    cin>>v_puan;
    cout<<"Sinav final puanini giriniz : "<<endl;
    cin>>f_puan;
    ort = v_puan*0.4 + f_puan*0.6;
    cout<<"Ortalama : "<<ort<<endl;
}
```

Tip Dönüşümleri

Örnek:Aşağıdaki programı izleyerek ekran çıktısını belirtin ve bellek değişkenlerinin değişimini gösterin.

```
#include <iostream>
using namespace std;

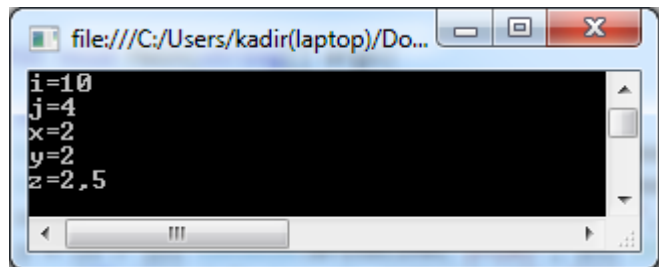
int main(){
    int a, b; double ort;
    cout<<"a : ";
    cin>>a;
    cout<<"b : ";
    cin>>b;
    ort = (a + b) / 2;
    cout<<"ortalama : "<<ort<<endl;
    ort = (a + b) / 2.0;
    cout<<"ortalama : "<<ort<<endl;}
```



Örnek:

```
#include <iostream>
using namespace std;

int main(){
    float x, y, z; int i, j;
    cout<<"i : ";
    cin>>i;
    cout<<"j : ";
    cin>>j;
    cout<<"i = "<<i<<endl<<"j = "<<j<<endl;
    x = float(i/j);
    y=i/j;
    z=float(i)/j;
```



```
cout<<"x = "<<x<<endl<<"y = "<<y<<endl<<"z = "<<z<<endl;
```

```
}
```

Örnek:

```
#include <iostream>
```

```
using namespace std;
```

```
int main(){
```

```
    int x = 0, y = 2, z = 1025;
```

```
    float a = 0.0f, b = 3.14159f, c = -37.234f;
```

```
    /* Arttırma */
```

```
    x = x + 1; /* Bu x i bir arttırır */
```

```
    x++; /* Bu da.. */
```

```
    ++x; /* Bu da.. */
```

```
    z = y++; /* z = 2, y = 3 */
```

```
    cout<<"z : "<<z<<"    "<<"y : "<<y<<endl;
```

```
    z = ++y; /* z = 4, y = 4 */
```

```
    cout<<"z : "<<z<<"    "<<"y : "<<y<<endl;
```

```
    /* Azaltma */
```

```
    y = y - 1; /* Bu y nin degerini bir azaltır */
```

```
    y--; /* Bu da.. */
```

```
    --y; /* Bu da.. */
```

```
    y = 3;
```

```
    z = y--; /* z = 3, y = 2 */
```

```
    cout<<"z : "<<z<<endl;
```

```
    z = --y; /* z = 1, y = 1 */
```

```
    cout<<"z : "<<z<<endl;
```

```
    /* aritmetik islemler */
```

```
    a = a + 12; /* a ya 12 eklemek */
```

```
    a += 12; /* 12 daha eklemek.. */
```

```
    cout<<"a : "<<a<<endl;
```

```
    a *= 3.2f; /* a yi 3.2 ile carpmak */
```

```
    a -= b; /* b yi a dan cikarmak */
```

```
    cout<<"a : "<<a<<endl;
```

```
    a /= 10.0f; /* a yi ona bolmek */
```

```
    cout<<"a : "<<a<<endl;
```

```
    //-----sartli islemler-----
```

```
    a = (b >= 3.0f ? 2.0f : 10.5f); /* Bu islem..... */
```

```
    if (b >= 3.0) /* ve bu islemler.. */
```

```
        a = 2.0f; /* birbiri ile aynidir */
```

```
    else /* ve ayni sonucu */
```

```
        a = 10.5f; /* saglarlar. */
```

```
    cout<<"a : "<<a<<endl;
```

```
    //-----
```

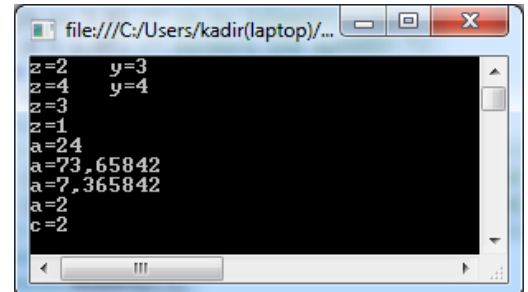
```
    c = (a > b ? a : b); /* c, a yada b nin max ini alır */
```

```
    c = (a > b ? b : a); /* c, a yada b nin min ini alır. */
```

```
    //-----
```

```
    cout<<"c : "<<c<<endl;
```

```
}
```



S.6.

```
#include <iostream>
using namespace std;
int main(){
    int x=10,y=5;
    cout<<++x-y--<<endl;
    cout<< x+++--y <<endl;
    cout<<x++<<endl;
    cout<<++x+y--<<endl;
    cout<<x-y<<endl;
}
```

S.7.

```
#include <iostream>
using namespace std;
int main(){
    int x=7,y=5,z;
    z = x++ + y;
    y = z-- - x;
    x = y++ + z;
    cout<<"x = "<<x<<endl;
    cout<<"y = "<<y<<endl;
    cout<<"z = "<<z<<endl;
}
```

KOŞUL DEYİMLERİ

Birkaç seçenektan birini seçmek veya bir deyimin bir koşula bağılı olarak işlemek için kullanılır. Koşul, programın akışını değıştiren bir yapıdır. Oldukça basit bir yapı olmasına karşın doğru yerlerde kullanabilmek için iyi bilinmesi ve çok pratik yapılması gerekmektedir.

IF-Then-Else Deyimi

```
if (<mantıksal ifade veya koşul>)
{ blok_doğru;}
else
{ blok_yanlış;}
```

Mantıksal ifade doğru ise blok_doğru, yanlış ise else sözcüğünden sonraki blok_yanlış yürütülür. else kısmı seçimlidir, gerekmiyorsa kullanılmayabilir. Yapılacak işlem tek bir işlem ise blok açıp kapatmaya gerek yoktur.

Örnek: Girilen sayının tek/çift olduğunu yazan programı yazın.

```
#include <iostream>
using namespace std;
int main()
{ setlocale(LC_ALL,"Turkish"); //Türkçe karakter
  int sayi;
  cout<<"Sayıyı Girin : ";
  cin>>sayi;
  if(sayi%2==0)
  { cout<<"Girilen sayı çift"; }
```

```
else {cout<<"Girilen sayı tek"; }  
}
```

Ödev: Girilen sayının tek/çift olduğunu yazan programı % (mod alma) kullanarak tekrar yazınız.

Örnek:

```
#include <iostream>  
using namespace std;  
int main()  
{  
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter  
    int yil, subat, gunyil;  
    cout<<"Yıl değerini giriniz";  
    cin>>yil;  
    if (yil % 4 == 0)  
    {  
        subat = 29;  
        gunyil = 366;  
    }  
    else  
    {  
        subat = 28;  
        gunyil = 365;  
    }  
    cout<<yil<<" yılı "<<gunyil<<" gündür"<<endl;  
    cout<<yil<<" yılı Şubat Ayı "<<subat<<" gündür"<<endl;  
}
```

Örnek:

```
#include <iostream>  
using namespace std;  
int main()  
{  
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter  
    int s1, s2, s3,s4, ek;  
    cout<<"s1 : ";  
    cin>>s1;  
    cout<<"s2 : ";  
    cin>>s2;  
    cout<<"s3 : ";  
    cin>>s3;  
    cout<<"s4 : ";  
    cin>>s4;  
    ek = s1;  
    if (ek > s2) ek = s2;  
    if (ek > s3) ek = s3;  
    if (ek > s4) ek = s4;  
    cout<<"En küçük olanı " <<ek<<endl;  
}
```

IF-ELSE IF YAPISI

```

If (koşul1)
    { Blok_A }
Else If (koşul2)
    { Blok_B }
Else If (koşul3)
    { Blok_C }
Else
    { Blok_D }
...

```

IF-ELSE IF yapısında koşul1 sağlanırsa Blok_A yapılır sağlanmazsa koşul2'ye bakılır ve sağlanırsa Blok_B yapılır ve bu şekilde devam eder. Hiçbir koşul sağlanmazsa Else'den sonra gelen Blok_D'deki işlemler yapılır. Normal If yapısında olduğu gibi bu yapıda da yapılacak işlem tek bir işlem ise blok açıp kapatmaya ({}) gerek yoktur.

Örnek:

```

#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    float fat_top, vergi_iade;
    cout<<"Fatura toplamalarını giriniz: ";
    cin>>fat_top;
    if (fat_top>0)
    {
        if (fat_top < 60000)
            vergi_iade = fat_top * 0.10f;
        else if (fat_top < 120000)
            vergi_iade = 6000 + (fat_top - 60000) * 0.20f;
        else if (fat_top < 200000)
            vergi_iade = 18000 + (fat_top - 120000) * 0.12f;
        else
            vergi_iade = 27600 + (fat_top - 200000) * 0.05f;
        cout<<"Ödenecek vergi iadesi = "<<vergi_iade<<endl;
    }
    else cout<<"Negatif değer girdiniz ";
}

```

Örnek:

```

#include <iostream>
using namespace std;
int sayi=55;
int main()
{ if (sayi<10)
    cout<<"Koşul Doğru"<<endl;
  else
    cout<<"Kosul Yanlis"<<endl;
  system("PAUSE");
  return 0;
}

```

```
}
```

Örnek:

```
#include <iostream>
using namespace std;
int main()
{ int s1,s2;
  cout<<"s1=";
  cin>>s1;
  cout<<"s2=";
  cin>>s2;
  if (s1>0 || s2>0)
    cout<<"En az biri
pozitif"<<endl;
  else
    cout<<"Ikisi de
negatif"<<endl;
  system("PAUSE");
  return 0;
}
```

```
s1=5
s2=-5
En az biri pozitif
Devam etmek için bir tuşa basın . . .
```

```
s1=-5
s2=-7
Ikisi de negatif
Devam etmek için bir tuşa basın . . .
```

Örnek:

#include <iostream>

```
using namespace std;
int main()
{ double x=7;int y=3;
  if (x==0)
    if (y==2) x-=3;
  else x=(x-7)/2; //hangi IF deyimine ait
  cout<<"x="<<x<<"\n";
  system("PAUSE");
  return 0; }
```

```
x=7
Devam etmek için bir tuşa basın . . .
```

switch Devimi

```
switch(değişken) {
  case seçenek1 :
    Yapılacak İşlemler;
    break;
  case seçenek2 :
    Yapılacak İşlemler;
    break;
  .
  .
  default :
    Yapılacak İşlemler;
    break;
}
```

Değişkenin aldığı değere eşit seçeneğin olup olmadığına bakar. Var ise **o noktadan sonraki deyimler yürütülür**. **switch** deyiminin sonuna gelindiğinde veya **break** deyimini ile karşılaşıldığında yürütme işlemi durur ve programın akışı switch deyimini izleyen deyim ile devam eder. default kısmı seçimliktir. Seçeneklerin hiçbiri uygun değil ise yürütülür. Switch deyimini yerine if deyimini kullanılabilir. Ancak switch deyimini programı daha okunabilir kıldığı için gerekli olduğu durumlarda kullanılmalıdır.

Örnek:

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "Turkish"); //Türkçe karakter
    int i = 5;
    switch (i)
    {
        case 1: cout<<"Bir"<<endl; break;
        case 2: cout<<"İki"<<endl; break;
        default:cout<<"Hiçbiri"<<endl; break;
    }
}
```

Örnek:

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "Turkish"); //Türkçe karakter
    char islem; int s1, s2, s3;
    cout<<"sayi1 : ";
    cin>>s1;
    cout<<"sayi2 : ";
    cin>>s2;
    cout<<"İşlem Girin(+,-,*,/)";
    cin>>islem;

    switch (islem)
    {
        case '+': s3 = s1 + s2; break;
        case '-': s3 = s1 - s2; break;
        case '*': s3 = s1 * s2; break;
        case '/': s3 = s1 / s2; break;
        default: s3 = 0; cout<<"Hatalı işlem"<<endl; break;
    }
    cout<<"Sonuç = "<<s3;
}
```


Örnek: Mevsimleri yazan program.

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    short ay;
    cout<<"Kaçınıcı Ay: ";
    cin>>ay;
    switch (ay) {
        case 3:
        case 4:
        case 5: cout<<"ilkbahar"<<endl; break;
        case 6:
        case 7:
        case 8: cout<<"yaz"<<endl; break;
        case 9:
        case 10:
        case 11: cout<<"sonbahar"<<endl; break;
        case 12:
        case 1:
        case 2: cout<<"kış"<<endl; break;
    }
}
```

Konu Sonu Çalışma Soruları

S. 1. Aşağıda verilen programları izleyin ve ekran çıktılarını yazın?

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int x = 0, y = 1, z = 1;
    if (x == 0) y = 0;
    if (y == 0) z = 0;
    if (z == 1) x = 1;
    cout<<"x = "<<x<<endl;
    cout<<"y = "<<y<<endl;
    cout<<"z = "<<z<<endl;
}
```

S. 2.

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
```

```

int x=0, y = 1, z=0;
if (y==1) x = 5;
cout<<"x = "<<x<<endl;
if (y!=1) x = 3;
else x = 5;
cout<<"x = "<<x<<endl;
x = 1;
if (y < 0)
    if (y > 0) x = 3;
    else x = 5; //hangi if deyimine ait
cout<<"x = "<<x<<endl;
if (y > 0) { z = y; x = 3; }
else if (y == 0) x = 5;
else x = 7;
cout<<"x = "<<x<<endl;
cout<<"z = "<<z<<endl;

if (!(x == z)&(z == y)) x = 13;
cout<<"x = "<<x<<endl;
cout<<"z = "<<z<<endl;
}

```

S. 3.

```

#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int i = 6;
    //&& birinci false ise 2.'ye bakmaz.
    //|| birinci true ise 2.ye bakmaz.
    if((++i<7)&&(i++/6==0) || (++i<=9));
    cout<<i<<endl;
    i = 6;
    //& ve | her türlü durumda bütün koşullara bakar
    if((++i<7) & (i++/6==0) | (++i<=9));
    cout<<i<<endl;
}

```

Soru 4. Sınav notunu harfe dönüştüren programı yazınız.

(>=90 :AA, 85-89:BA, 80-84:BB, 75-79:CB, 70-74:CC, 60-69:DC, <60 :FF)

```

#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int vize, final; float ort;
    cout<<"Vize Değeri=";
    cin>>vize;
    cout<<"Final Değeri=";
    cin>>final;
    ort = vize * 0.4f + final * 0.6f;
}

```

```

    cout<<ort<<endl;
    if (ort >= 90) cout<<"-->Harf Notu:AA"<<endl;
    else if (ort >= 85) cout<<"-->Harf Notu:BA"<<endl;
    else if (ort >= 80) cout<<"-->Harf Notu:BB"<<endl;
    else if (ort >= 75) cout<<"-->Harf Notu:CB"<<endl;
    else if (ort >= 70) cout<<"-->Harf Notu:CC"<<endl;
    else if (ort >= 60) cout<<"-->Harf Notu:DC"<<endl;
    else cout<<"-->Harf Notu:FF"<<endl;
}

```

S.5. Aşağıdaki programı izleyerek ekran çıktısını yazın.

```

#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    double x=5;int y=2;
    if (x==0)
    if (y==2) x-=3;
    else x=(x-5)/2;
    cout<<"x: "<<x;
}

```

DÖNGÜ DEYİMLERİ

FOR DEYİMİ

Bir işlem birden fazla yapılmak isteniyorsa mutlaka döngü kullanılmalıdır. FOR döngü için en çok kullanılan deyimdir.

for (i=başlangıç değeri ; i< bitiş değeri; Artış miktarı)

```

{
    Kodlar;
}

```

For döngüsünde döngü değişkeni(i) başlangıç değeri ile başlar ve döngü içindeki kodlar gerçekleştirilir. Döngünün her adımında döngü değişkeninin değeri artış miktarı kadar artar ve bitiş koşuluna ulaşıp ulaşılmadığına bakılır. Ulaşılmamışsa döngü içindeki kodlar tekrar yapılır. Ulaşılmışsa döngü dışına çıkılır.

Örnek:

```

#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    short i;
    for (i = 1; i < 5; i++)
    {cout<<i<<endl;                //dikkat blok yok
      cout<<++i<<endl;}
}

```

Örnek: Aşağıdaki programı izleyerek ne iş yaptığını belirtin.

```
#include <iostream>
using namespace std;
int main()
{short i;
    for (i = 1; ; i++)
        cout<<i<<endl;
}
```

Örnek:

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    double fakt = 1; int j, i;
    cout<<"Sayı= "; cin>>i;
    for (j = 1; j <= i; j++)
        fakt = fakt * j;
    cout<<"Faktöriyel="<<fakt<<endl;
}
```

Örnek:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int i,sayi;
    for (i=1;i<=10;i++)
    { sayi = (rand() % 100) + 1; //1-100 arası rasgele sayı üret
      cout<<i<<" Sayi="<<sayi<<endl;
    }
}
```

Örnek:

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int adet = 0;
    for (int i = 0; i <= 30; i += 5)
    {
        if (adet % 3 == 0)
            cout<<endl;
        cout<<"\t"<<i;

        adet++;
    }
}
```

```
}
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{ for (int i=10;i<10;i++)
  printf("%d\n",i);
  printf("İşlem Bitti");
  getch();
}
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{
  int i;
  clrscr();
  for (i = 1; i < 5; i++)
    printf("%d ",i);
    printf("\n%d",i);
    getch();
}
```

Break ve Continue Deyimleri

- Bazı uygulamalarda, döngü işlemi tamamlanmadan döngünün sona erdirilmesi söz konusu olabilir.
- Bu gibi durumlarda break; deyimi kullanılır.
- Döngü içinde bu deyimle sıra geldiğinde, break ardından döngü sonuna kadar olan tüm deyimler atlanır ve programa döngü dışındaki bir sonraki adımdan itibaren devam edilir.
- Bir döngüyü terk etmeden bir adımının atlanması söz konusu ise, continue; deyimi kullanılır.
- Bu deyim döngünün işlemlerini sona erdirmez, sadece bir sonraki döngü adımına geçilmesini sağlar.
- Eğer for döngüsü kullanılıyorsa, işlem sırası bu deyimle geldiğinde, bu deyimden döngü sonuna kadar olan deyimler çalışmaz, döngü bir artırılarak sonraki döngüye geçilir.

Örnek:

```
#include <iostream>
using namespace std;
int main()
{ for (int i=1;i<100;i++)
  { cout<<i<<"\n";
    if (i==5) break;
  }
  system("PAUSE"); return 0;}
```



```
1
2
3
4
5
Devam etmek i
```

Örnek:

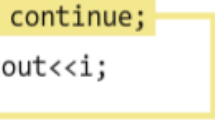
```
#include <iostream>
using namespace std;
int main()
{ for (int i=1;i<15;i++)
    { if (i>8 && i<12) continue;
      cout<<i<<"\n";
    }
  system("PAUSE");
  return 0;
}
```

```
1
2
3
4
5
6
7
8
12
13
14
Devam etmek için
```

Örnek:Aşağıdaki programları son=5 için izlevin.

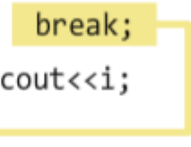
continue deyiminin while döngüsünde kullanımı:

```
...
while(i<=son)
{
  if(i==3)
    continue;
  cout<<i;
}
cout<<"\n";
```



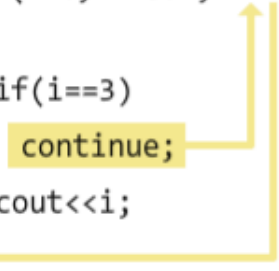
break deyiminin for döngüsünde kullanımı:

```
...
for(i=0; i<son; i++)
{
  if(i==3)
    break;
  cout<<i;
}
cout<<"\n";
...
```



continue deyiminin for döngüsünde kullanımı:

```
...
for(i=0; i<son; i++)
{
  if(i==3)
    continue;
  cout<<i;
}
cout<<"\n";
...
```



İç içe Döngü Örneği:

Çarpım Tablosunu oluşturup ekranda gösteren programı yazınız.

Çözüm:

İç içe for döngülerine en güzel örneklerden biri çarpım tablosudur. İçteki döngünün her tamamlanışından sonra dıştaki döngü otomatik olarak bir artar. Dıştaki döngü değişkeni $i=1$ iken içteki döngü değişkeni j , 1 den 10 a kadar değer alır. j en son değerine ulaştıktan sonra ($j=10$ olduktan sonra) i değişkeni 2 değerini alır. $i=2$ iken j değişkeni tekrar 1 den 10 a kadar değer alır ve bu işlem 10 kez tekrarlanır. Yani toplam 100 kez çarpma işlemi gerçekleşir.

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int i, j;
    for (i = 1; i <= 10; i++)
    {
        for (j = 1; j <= 10; j++)
            cout<<"\t"<<i * j;
        cout<<endl;
    }
}
```

Ödev:

1' den 1000'e kadar olan sayılar içerisinde 5'e tam bölünen aynı zamanda 7' ye tam bölünemeyen sayıları sayan, toplamlarını hesaplayan ve bu sayıları listeleyen bir program yazınız.

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{ //birden fazla sayının faktöryelini hesaplamak için iç içe döngü
  kullanımı
    double fakt =1;int sayi;int adet;
    printf("Kac adet faktoryel degeri hesaplanacak:");
    scanf("%d",&adet);
    for(int k=0;k<adet;k++)
    {
        printf("\n%d.Sayiyi Girin=",k+1);
        scanf("%d",&sayi);
        fakt=1;
        for (int i=1; i<=sayi; i++)
            fakt =fakt*i;
        printf("Faktoryel  =%7.0f\n",fakt);
    }
    getch();
}
```

```
Kac adet faktoryel degeri hesaplanacak:3
1.Sayiyi Girin=5
Faktoryel  = 120
2.Sayiyi Girin=3
Faktoryel  = 6
3.Sayiyi Girin=10
Faktoryel  =3628800
```

Örnek:

0 ile 100 arasında klavyeden girilen 10 notun en büyüğünü, en küçüğünü hesaplayan ve notların ortalamasını bulan program.

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int eb=0, ek=0, toplam = 0;
    int adet = 5; int notu;
    cout<<"Kaç Adet Not Gireceksiniz:";
    cin>>adet;
    for (int i = 0; i < adet; )
    {   cout<<i+1<<". Notu Giriniz:"<<endl; cin>>notu;
        if (notu < 0 || notu > 100)
            cout<<"Yanlış Not...";
        else
        {   if (i == 0) { eb = notu; ek = notu; }
            if (notu > eb)
                eb = notu;
            if (notu < ek)
                ek = notu;
            toplam += notu;
            i++;
        }
    }
    cout<<"=====Sonuçlar Yazdırılıyor===== "<<endl;
    cout<<"En yüksek not="<<eb<<endl;
    cout<<"En düşük not="<<ek<<endl;
    float ortalama = (float)toplam / (float)adet;
    cout<<"Ortalama="<<ortalama<<endl;
}
```

Örnek:

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int tektoplam = 0, cifttoplam = 0; int i;
    for (i = 0; i <= 100; i++)
    {
        if (i % 2 == 1) tektoplam += i;
        else cifttoplam += i;
        if (i == 10) break;
    }
    cout<<"Tek Sayıların Toplamı = "<<tektoplam<<endl;
    cout<<"Çift Sayıların Toplamı = "<<cifttoplam<<endl;
    cout<<"i : "<<i<<endl; }
```


Örnek: Aşağıda verilen C programının ne iş yaptığını bir cümle ile belirtin?

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    char i;
    for (i = 'a'; i <= 'z'; i++)
    {
        cout<<i<<endl;
    }
}
```

Örnek: Aşağıda verilen C programının ne iş yaptığını bir cümle ile belirtin?

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    //-----1. yöntem
    int sayi = 5;
    for(;;)
    {
        cout<<sayi % 2;
        sayi = sayi / 2;
        if (sayi == 0) break;
    }
    //cout<<sayi / 2;
}
```

Örnek: Aşağıda verilen C programının ne iş yaptığını bir cümle ile belirtin?

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    //-----1. yöntem
    int sayi = 5;
    for(int i=0;i<=0;)
        cout<<++i;
    cout<<++i;
}
```

WHILE DEYİMİ

Mantıksal ifade doğru olduğu sürece döngü içindeki yapılacak işlemler yürütülür. Eğer yanlış ise kontrol döngü dışındaki bir sonraki deyim'e geçer. Şarta bağlı olarak tek deyim kullanılabileceği gibi blok içine alınarak birden fazla deyimde yürütülebilir.

C'de While komutunun iki farklı kullanımı vardır.

1. Kullanım: Bu kullanımda döngü içindeki deyimlerin şarta bağlı olarak hiç yapılmama ihtimali de vardır.

```
while <koşul>
{
    <Koşula bağlı olarak yapılacak işlemler>
}
```

2. Kullanım: Her şartta döngü içindeki deyimler en az 1 kere yapılır.

```
do{
    <Koşula bağlı olarak yapılacak işlemler>
}while <koşul>
```

Örnek: Yeni kiraladığımız bir evin kirasının 1000 TL olduğunu varsayalım. Ve her yıl kirayı %5 arttıracığımızı düşünelim. Önümüzdeki 10 yıl içinde vereceğimiz kira miktarının ne olacağını hesaplayan bir program yazalım. Birinci yazım döngü kullanmadan gerçekleştirilirken 2. Yazım döngü ile yapılmıştır. İki örneği inceleyerek neden döngü kullanılması gerektiğini belirtin ve anlamaya çalışın.

```
#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL, "Turkish"); //Türkçe karakter
    int Kira = 1000, ArtisMiktari = 5;
    cout<<"2006 = "<<round(Kira)<<" TL"<<endl;
    cout<<"2007 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2008 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2009 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2010 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2011 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2012 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2013 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2014 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2015 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
    cout<<"2016 = "<<round(Kira * (1 + ArtisMiktari / 100))<<" TL"<<endl;
}
```

Örnek:

```
#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int BaslangicYili = 2006; int BitisYili = 2016;
    int Kira = 1000; int ArtisMiktari = 5;
    while (BaslangicYili <= BitisYili)
    {
        cout<<BaslangicYili<<" "<<Kira<<" YTL"<<endl;
        Kira *= (1 + ArtisMiktari / 100);
        Kira = round(Kira);
        BaslangicYili++;
    }
}
```

İpucu: Birinci örnekte 10 yıl için çözüm sağlanabilir. Ancak yıl sayısı 100'e ya da 1000'e çıkarsa ne yaparsınız.

Örnek: Aşağıdaki iki farklı programı izleyerek ne iş yaptığını belirtin ve karşılaştırın.

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

int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int a, b, c;
    cout<<"a = "; cin>>a;
    cout<<"b = "; cin>>b;
    c = 0;
    while (b > 0) {
        c = c + a;
        b = b - 1;
    }
    cout<<"Sonuç = "<<c;
}
```

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

int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int a, b, c;
    cout<<"a = "; cin>>a;
    cout<<"b = "; cin>>b;
    c = 0;
    do{
        c = c + a;
        b = b - 1;
    } while (b > 0);
}
```

```

        cout<<"Sonuç = "<<c;
    }

```

Örnek: Aşağıdaki programı izleyerek ne iş yaptığını belirtin.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int n; double f;
    cout<<"Sayı giriniz="; cin>>n;
    f = 1;
    while (n > 1) {
        f = f * n;
        n = n - 1;
    }
    cout<<"Sonuç = "<<f;
}

```

Örnek: Klavyeden girilen sayıları okuyup sayıların toplamı 21'den büyük veya eşit olduğu zaman duran C programını yazın.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int i, toplam = 0;
    while (toplam < 21)
    {
        cin>>i;
        toplam+= i;
    }
    cout<<"Toplam = "<<toplam;
}

```

Örnek:

1993 yılı itibarı ile ülke nüfusu 6000. Yıllık nüfus artış oranı %2.3 tür. Sonraki 10 yılda ülke nüfusunu yıllara göre listeleyen program.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int i; /* sayac */
    int yil; /* yillar */
    float nufus; /* nufus miktarı */
    float artis; /* artis oranı */
    artis = 2.3;
    yil = 1993;
    nufus = 6000;
    cout<<yil<<" "<<nufus<<endl;
    i = 1;
    do
    {
        nufus += nufus * (artis) / 100;
    }
}

```

```

        cout<<yil+i<<" "<<nufus<<endl;
        i += 1;
    } while (i < 11); }

```

Örnek: Aşağıdaki programı izleyerek değişken değişimlerini gösterin ve ne iş yaptığını bir cümle ile belirtin. Ayrıca programı yeniden düzenleyerek not değerlerinin 0-100 aralığında girilmesi için gerekli kontrolleri uygun yere yerleştirin.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int ogr_say, not_top, not1=0;
    float ort=0;
    not_top = 0;ogr_say=0;
    while (true) {
        cout<<"Sıradaki öğrencinin notu = ";
        cin>>not1;
        if (not1 < 0) break;
        else
        {
            not_top = not_top + not1;
            ogr_say++;
        }
    }
    ort=(float) not_top / ogr_say;
    cout<<"Ortalama = "<<round(ort);
}

```

Örnek: Öyle bir C programı yazın ki program pin kodu istesin. Pin kodunu doğru girmek için 3 hak versin. Her yanlış girişte “Tekrar deneyiniz” mesajını, hak bittiğinde “Uygulamanız bloke olmuştur” mesajını, doğru giriş yapıldığında ise “HOŞ GELDİNİZ” mesajını versin. Her mesajdan sonrada mutlaka kalan hak da kullanıcıya bildirilsin.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int hak = 3, pin, kod;
    kod = 4402;
    while (hak > 0)
    { cout<<"\nPin kodunu giriniz:";
      cin>>pin;
      hak-=1;
      if (pin == kod)
      { cout<<"HOŞ GELDİNİZ...";
        break;
      }
      else
      {
          if (hak == 0)
              cout<<"Uygulamanız bloke olmuştur.";
          else
              cout<<"Tekrar Deneyiniz";
      }
    }
}

```

```

    }
    cout<<"Kalan hak:"<<hak;
    }
}

```

Örnek: 10'luk sayı sisteminden 2'lik sayı sistemine farklı yöntemlerle dönüşüm.

```

#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    //-----1. yöntem
    int sayi = 5;
    while (sayi > 0)
    {
        cout<<sayi % 2;
        sayi = sayi / 2;
    }
    //-----2.yöntem
    sayi = 5; cout<<endl;
    for (int i = 0; sayi > 0; i++)
    {
        cout<<sayi % 2;
        sayi = sayi / 2;
    }
    //-----3. Yöntem
    sayi = 5; cout<<endl;
    for(;;)
    {
        cout<<sayi % 2;
        sayi = sayi / 2;
        if (sayi == 0) break;
    }
    //Console.WriteLine(sayi / 2);
}

```

Konu Sonu Çalışma Soruları

S.1. Klavyeden girilen isim değerini klavyeden girilen sayı kadar ekrana yazdıran C programını yazın?

S.2. Klavyeden girilen 2 değer arasındaki çift sayıların toplamını hesaplayan C programını for döngüsü kullanarak yazın?

S.3. Klavyeden girilen sayının asal sayı olup olmadığını yazan C programını yazın?

S.4. Klavyeden girilen 2 değer arasındaki 5'e bölünebilen sayıların toplamını bulan C programını yazın?

S.5. Klavyeden girilen sayı kadar, klavyeden girilen sayıların ortalaması bulup ekrana yazdıran C programını yazın?

S.6. Dışarıdan girilen rastgele 10 tane sayıdan tek ve çift sayıların ortalamasını ayrı, ayrı bulup ekrana yazan C programını yazın?

S.7. İki sayının Ortak Bölenlerinin En Büyüğünü (OBEB) ve Ortak Katlarının En Küçüğünü (OKEK= $A \cdot B / \text{OBEB}$) bulan programı C’da yazınız.

S.8. 4 işlem yapan basit bir hesap makinesi programını C’da yazınız.

S.9. Dışardan iki sayı okuyup 1. sayıyı taban 2. sayıyı üs kabul ederek üs alma işlemini yapan programı C’da yazınız.

S.10. Dışardan okunan 10 tane rast gele sayıdan kaçının negatif kaçının pozitif olduğunu ve pozitifleri kendi arasında negatifleri kendi arasında toplayıp sonuçları ekrana yazan programı yazınız.

S.11. Sadece toplama işlemi kullanarak girilen iki sayıyı çarpan programı C’da yazınız.

S.12. Klavyeden girilen 10 sayı içerisinde a) 100-200 arasındaki sayıların adedini; b) 100’den küçük sayıların toplamını; c) 200’den büyük sayılardan da 4’e kalansız bölünebilenlerini ekrana yazdıran programı do-while döngüsü ve if komutlarıyla yazınız.

KONU SONU ÖDEVLERİ

ÖDEV: Toplama işlemini öğretmeye çalışan bir oyun programı yazılacaktır. Oyun başladığı zaman rastgele 2 tane 1-100 arasında sayı tutulacak, tutulan sayılar ekrana gösterilecek ve kullanıcıya bu sayıların toplamı nedir diye sorulacaktır. Eğer kullanıcı doğru cevap verirse “Tebrikler Bildiniz” değil ise “Üzgünüm Bilemediniz” diye mesaj verecektir. Her cevaptan sonra “Tekrar Oynamak istiyormusunuz(e/E)?” şeklinde bir soru sorulacak ve eğer kullanıcı “e” veya “E” ile karşılık verirse oyun tekrar başlayacaktır. Kullanıcının puanı her doğru cevap için 5 puan artacak, her yanlış cevap için ise 2 puan azalacaktır. Oyun sonlandığında kullanıcının verdiği doğru cevap sayısı, yanlış cevap sayısı ve puanı ekranda listelenmelidir. Bu işlemleri yapan programın C kodlarını yazınız.

ÖDEV: Bir otoparka park eden taksinin 1 saati 5TL, minibüsün 1 saati 6TL, ticari aracın 1 saati 6.5TL dir. Taksi 1 saatten sonraki her saat başı için %20 daha fazla, minibüs 1 saatten sonraki her saat başı için toplamda %21.5 ve ticari araç 1 saatten sonraki her saat başı için toplamda %25 daha fazla ödeme yapmaktadır. Buna göre klavyeden girilen araba tipi ve kalınan saat bilgisi girildikten sonra ekrana ödenecek otopark ücretini hesaplayan programın C kodlarını yazınız.

ÖDEV: Meteoroloji merkezi için bir program tasarlanması istenilmiştir. Programın çalışma şekli ise şöyle olmalıdır:

- İlk önce hangi ay için sıcaklık bilgisi girileceği kullanıcıya sorulacaktır.
- Girilen ay bilgisine uygun olarak o ayda kaç tane gün var ise kullanıcıdan gün sayısı kadar sıcaklık bilgisi girilmesi istenilecektir(gubat ayı için gün sayısını 28 alınız).
- Sıcaklık veri girişi bittikten sonra o ayın sıcaklık ortalaması ve en düşük sıcaklık bilgisi ekrana yazdırılacaktır. Bu işlemden sonra program sonlanacaktır.

Örnek Çıktı: Şubat Ayına ait Ortalama Sıcaklık=15,6 derecedir ve En düşük sıcaklık 6.Gün=10,1 derecedir.

ÖDEV: Bir komisyoncu sattığı mallardan fiyatı 50 TL kadar olanlardan %3, daha fazla olanlardan ise %2 komisyon almaktadır. Klavyeden girilen teker teker girilen 5 malın komisyonlarını bulup ekrana yazdıran ve en sonunda da toplam komisyonu ekrana yazdıran C programını yazınız.

ÖDEV: Kendisi hariç bütün pozitif çarpanları (tam bölenleri) toplamı, yine kendisine eşit olan sayılara “mükemmel sayı” denir. Örneğin $6=1+2+3$ ve $28=1+2+4+7+14$ gibi. Buna göre klavyeden girilen bir tamsayının “mükemmel sayı” olup olmadığını kontrol eden programın kodlarını yazınız.

DİZİLER

Aynı tipteki birden fazla değeri saklayabilen değişkenlerdir. Dizi değişkeninde değer sağlayabilmek için ya da dizi değişkeninden değer okuyabilmek için indis (index) numarası kullanılır. Dizideki ilk değerın indis numarası “1” değil “0” dır. Aynı özellikte olan ve aynı amaçla kullanılan bir grup verinin bilgisayar ortamında tutulduğu veri yapısıdır.

Saklanacak veriler çok fazla ise dizi kullanılır. Mesela 30 kişilik bir sınıfı isimlerini, notlarını almak istiyoruz. Bunun için değişken kullanmak istersek 30 isimler, 30’da notlar için 60 tane değişken kullanmamız gerekecek. Bunu bir okul için yaptığımızı düşünün. İçinden çıkılmaz bir hal alır. İşte bu gibi durumlar için dizi kullanıyoruz. Diziyi oluşturan veri kümeleri (elemanlar) hafızada bitişik olarak sıralı bir şekilde yerleşir.

C’de dizi aşağıdaki gibi tanımlanır.

değişken tipi değişken adı [];

Örnek : `int puan[10]; string ad[10];`

puan[0]	puan[1]	puan[2]	puan[3]	puan[4]	puan[5]	puan[6]	puan[7]	puan[8]	puan[9]

Bir dizi yukarıdaki şekilde düşünülebilir. Her bir eleman yukarıdaki gibi isimlendirilir. Bunun bize sağlayacağı avantaj indis değerlerinin değişkenlerle ifade edilebilmesidir. Bu indis değerleri vasıtasıyla dizi elemanlarına bir döngü içerisinde kolayca ulaşıp işlem yapılabilir. Yani tek bir satır komut ile tüm dizi elemanları ekranda yazdırılabilir.

Neden dizi kullanmamız gerektiğini bir örnekle açıklayalım.

Örnek: Sınıfımızda 40 kişinin isimlerini ve bilgisayar notlarını programa girip isteğe göre, en yüksek notu, notlara göre sıralamayı, ilk üç kişiyi, notların ortalamasını ve kaç kişinin kalıp geçtiğini bulmak istiyoruz. Bunun için 40 isim ve 40 bilgisayar notunu hafızada devamlı tutmamız gerekecektir. Değişken ve dizi yaklaşımıyla problemin çözümü aşağıdaki gibi olacaktır.

Aradaki Fark	Normal Değişken	Dizi
Tanımlama farkı	String isim1, isim2, isim3...isim40 İnt not1, not2,...not40	String isim[40] int notlar[40]
İsim ve notun klavyeden girilmesi	Oku isim1 Oku not1 Oku isim2 Oku not2 . . Oku isim40 Oku not40	For(i=0;i<40;i++) { Oku isim[i] Oku notlar[i] }
En büyük notu bulma	If (buyuk<not1) If(buyuk<not2) If(buyuk<not40)	for(i=0;i<40;i++) If (buyuk<notlar[i]) buyuk=notlar[i]

Aynı veri tipindeki birden fazla veriyi bellekte saklamak için kullanılır. Örneğin bir matrisin değerlerini bellekte tutmak için dizi kullanılmalıdır. Klavyeden girilen not değerlerinin sıralanıp listelenmesi isteniyorsa yine dizi kullanılmalıdır.

Örnek:

```
float a[100];
```

Bu tanımlama ile a isimli değişkeni 100 gerçel değer saklandığı bir diziyi gösterir. Bu 100 veriye a değişkeni ile erişilir.

Dizinin herhangi bir elemanına erişmek veya değiştirmek için kaçınca eleman olduğunu gösterir indis bilgisini vermek gerekir. **İlk elemanın indisi 0 dır.**

A[4] dizinin 5. elemanı

A[0] dizinin ilk elemanı

A[1] = 45; dizinin 2. elemanına 45 atanır

A[7] = A[7] + A[1]; dizinin 8. elemanına kendisi ile 2. elemanın toplamı atanır

Örnek:

1-10 arasındaki sayıların karesini dizi elemanlarına yükleyip istenen elemanı ekrana yazdıran C programını yazın.

```
#include<iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int a[10], eleman=1;
    for (int i=0; i<=9; i++)
        a[i] = (i+1)*(i+1);
    while (true)
    {
        cout<<"Kaçınca Eleman=";
        cin>>eleman;
        if (eleman < 1 || eleman > 10) break;
        cout<<"a["<<eleman<<"]="<<a[eleman-1]<<endl;
    }
}
```

Örnek: : Klavyeden 10 sayı oku. Tersten yazdır.

```
#include<iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int a[10], i;
    for (i=0; i<=9; i++)
    {
        cout<<i+1<<".sayıyı gir:";
        cin>>a[i];
    }
    cout<<"\n-----\n";
```

```

    for (i=9; i>=0; i--)
        cout<<i+1<<". sırada girilen sayı ="<<a[i]<<endl;
}

```

Örnek: Klavyeden girilen kelimeyi tersten yazdıran program.

```

#include <iostream>
#include<string.h>
using namespace std;
int main() {
    char kelime[10];int i,uzunluk;
    cout<<"Bir kelime girin=";
    gets(kelime);
    uzunluk=strlen(kelime);
    cout<<"Girdiğiniz Kelimenin Uzunluğu="<<uzunluk<<endl;
    cout<<"kelimenin tersten yazılışı=";
    for (i=uzunluk-1;i>=0;i--) cout<<kelime[i];
}

```

Ödev: Soldan sağa okunuşu ile sağdan sola okunuşu aynı olan kelimelere Palindrome kelimeler denir (kazak, kapak, iyi efe vb.). Girilen bir kelimenin Palindrome kelime olup olmadığını belirleyen C# programını yazın.

Ödev. Klavyeden 10 kişiye ait girilen vize ve final bilgilerine göre ortalamayı hesaplayıp ortalamaya göre küçükten büyüğe sıraladıktan sonra vize, final ve ortalama bilgilerini listeleyen C++ programını yazın?

Örnek: Klavyeden girilen 10 not değerini küçükten büyüğe sıralayıp ekrana yazan C programını yazın.

```

int not1[10]; int i,j,tut;
for (i=0;i<10;i++)
{cout<<i+1<<". Notu Girin=";
  cin>>not1[i];
}
for (i=0;i<9;i++)
{
    for (j=i+1;j<10;j++)
    {
        if (not1[i]>not1[j])
        {
            tut=not1[i];
            not1[i]=not1[j];
            not1[j]=tut;
        }
    }
}
cout<<"=====Sıralanmış Sayılar===== "<<endl;
for (i=0;i<10;i++)
    cout<<not1[i]<<endl;

```

ÇOK BOYUTLU DİZİLER

Tek Boyutlu Bir Dizi

6	34	23	-8	123	87	19	12
---	----	----	----	-----	----	----	----

↑
indeks[5]

2 Boyutlu(Düzenli) Bir Dizi

6	34	23	-8	123	87	19	12
2	43	3	22	-73	78	17	22
16	4	17	63	7	-31	-18	32
0	125	33	99	981	31	16	0

↑
indeks[3][2]

↓
index[0][5]

←
indeks[1][6]

2 Boyutlu(Düzensiz) Bir Dizi

6	34	23	-8	123	87	19	12			
2	43	3	22	-73	78	17	22	9	255	127
16	4	17	63	7	-31	-18	32	8		
0	125	33	99	981	31	16				

↑
indeks[3][2]

↓
indeks[0][5]

↓
indeks[1][9]

←
indeks[2][7]

3 Boyutlu Bir Dizi

12	120	-87	4	334	619
0	212	67	33	21	7
176	56	8	134	75	-17

↑
indeks[2][0][1]

↑
indeks[2][1][2]

↓
indeks[0][0][0]

↓
indeks[0][1][0]

←
indeks[1][1][1]

Diziler, birden fazla boyut olabilir. Örneğin, aşağıdaki bildirim, dört satır ve iki sütundan oluşan iki boyutlu bir dizi oluşturur.

```
int array[4][2]; //iki boyutlu dizi
```

```
int array1[4][2][3]; //üç boyutlu dizi
```

Örnek:

```
#include<iostream>
using namespace std;
int main()
{setlocale(LC_ALL,"Turkish"); //Türkçe karakter
  int array2Da[4][2]= { { 1, 2 }, {3,4}, {5,6}, {7, 8 } };
  // A similar array with string elements.
  string array2Db[3][2]={{"one","two"},"three","four" },
                        { "five", "six" } };
  // The same array with dimensions specified.
  int array3Da[2][2][3]= { { { 1, 2, 3 }, { 4, 5, 6 } },
                          { { 7, 8, 9 }, { 10, 11, 12 } } };

  // Accessing array elements.
  cout<<array2Da[0][0]<<endl;
  cout<<array2Da[0][1]<<endl;
  cout<<array2Da[1][0]<<endl;
  cout<<array2Da[1][1]<<endl;
  cout<<array2Da[3][0]<<endl;
  cout<<array2Db[1][0]<<endl;
  cout<<array3Da[1][0][1]<<endl;
  cout<<array3Da[1][1][2]<<endl;
} // Output:
// 1
// 2
// 3
// 4
// 7
// three
// 8
// 12
```

Örnek:

```
#include<iostream>
using namespace std;
int main()
{
  setlocale(LC_ALL,"Turkish"); //Türkçe karakter
  int a[2][2]; //iki boyutlu dizi
  a[0][0] = 1; a[0][1] = 2;
  a[1][0] = 3; a[1][1] = 4;

  for (int i = 0; i < 2; i++)
  {
    for (int j = 0; j < 2; j++)
    {
      cout<<a[i][j]<<"\t";
    }
    cout<<endl;
  }
}
```

Örnek: Aşağıdaki örnekte matrisin elemanlarının nasıl girildiğini inceleyin.

```
#include<iostream>
#include "windows.h"
using namespace std;
void git ( short x, short y )
{
    COORD coord = {x, y};
    SetConsoleCursorPosition ( GetStdHandle ( STD_OUTPUT_HANDLE ), coord );
}
int main()
{
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int x = 4, y = 4;
    int total=0, column=0, row=0;
    cout<<"\n\niki boyutlu dizinin elemanlarını enter tuşuna basarak
sırayla giriniz:\n\n";
    //iki boyutlu bir dizi tanımla
    int ikiliDizi[x][y];
    //iki boyutlu diziyi console dan girilen sayılar ile doldurma
    for (int i = 0; i < x; i++)
    {
        for (int j = 0; j < y; j++)
        {
            //konsole ekranında cursor ın yerini belirleme
            //böylece matris yapısı oluşturabilme
            gotoxy((j + 1) * 5, i + 5);
            //iki boyutlu matrisi diziye aktarma
            cin>>ikiliDizi[i][j];
            total++;
        }
        row++;
    }

    //matrisin özelliklerini console ekranına yazdırma
    int total1 = sizeof(ikiliDizi)/sizeof(ikiliDizi[0][0]);
    int column1 = sizeof(ikiliDizi[0])/sizeof(ikiliDizi[0][0]);
    int row1 = sizeof(ikiliDizi)/sizeof(ikiliDizi[0]);

    cout<<endl;
    cout<<"Satır Sayısı= "<<row1<<endl;
    cout<<"Sütun Sayısı= "<<column1<<endl;
    cout<<"Eleman Sayısı= "<<total1<<endl;

    cout<<endl;
    cout<<"Satır Sayısı= "<<row<<endl;
    cout<<"Sütun Sayısı= "<<total/row<<endl;
    cout<<"Eleman Sayısı= "<<total<<endl;
}
```

Örnek:

```

#include<iostream>
using namespace std;
int main()
{int a[3][3];
    int b[3][3];
    int c[3][3];
    int i, j;
    for (i = 0; i <=2; i++)
    {
        for (j = 0; j <= 2; j++)
        {
            cout<<"a["<<i<<"]["<<j<<"]="";
            cin>>a[i][j];
            cout<<"b["<<i<<"]["<<j<<"]="";
            cin>>b[i][j];
        }
    }
    cout<<"======"<<endl;
    for (i = 0; i <= 2; i++)
    {
        for (j = 0; j <= 2; j++)
        {
            c[i][j] = a[i][j] + b[i][j];
            cout<<c[i][j]<<"\t";
        }
        cout<<endl;
    }
}

```

Ödev: Klavyeden girilen 3x2 ve 2x4'lük matrisleri çarpıp sonucu ekrana matris formatında yazan C programını yazın?

Örnek: Diziyi tersine çevirme.

```

#include <iostream>
#include <string>
using namespace std;
int main()
{ int dizi[6] = { 1, 2, 3, 4, 5, 6 };
    int elemansayisi=sizeof(dizi)/sizeof(dizi[0]);
    int tersDizi[elemansayisi];
    cout<<"Eleman Sayısı="<<elemansayisi<<endl;
    // dizinin tersini for döngüsü kullanarak al
    for (int i = 0; i < elemansayisi; i++)
        tersDizi[i] = dizi[elemansayisi - 1 - i];

    cout<<"Ters Dizi=";
    for (int i = 0; i < elemansayisi; i++)
    {
        cout<<tersDizi[i];
        cout<<" ";
    }
    cout<<endl;  system("PAUSE");  return 0;}

```

Örnek:

```
#include<iostream>
using namespace std;
int main()
{setlocale(LC_ALL,"Turkish"); //Türkçe karakter
int maxOgrenciSayisi = 100;
    int kayitOlacakOgrenciSayisi;
    string tcNo[maxOgrenciSayisi];
    string ad[maxOgrenciSayisi];
    string soyad[maxOgrenciSayisi];
    string ogrenciNo[maxOgrenciSayisi];
    cout<<"Kaç öğrenci kayıt edeceksiniz: ";
    cin>>kayitOlacakOgrenciSayisi;
    for (int i = 0; i < kayitOlacakOgrenciSayisi; i++)
    {
        cout<<i+1<<". öğrencinin ismini girin: ";
        cin>>ad[i];
        cout<<i + 1<<". öğrencinin soyismini girin: ";
        cin>>soyad[i];
        cout<<i + 1<<". öğrencinin TC Kimlik numarası: ";
        cin>>tcNo[i];
        cout<<i + 1<<". öğrencinin okul numarası: ";
        cin>>ogrenciNo[i];
    }
    cout<<"====Kayıtlar Listeleniyor===="<<endl;
    cout<<" Tc Kimlik\tÖğrenci No\tAdı\tSoyadı"<<endl;
    cout<<"======"<<endl;

    for (int i = 0; i <= kayitOlacakOgrenciSayisi;i++ )
    cout<<tcNo[i]<<"\t\t"<<ogrenciNo[i]<<"\t\t"<<ad[i]<<"\t\t"<<soyad[i]<<endl;
    }
```

Örnek:

Kullanıcı tarafından girilen bir cümlede sesli harflerin sayısını bulan C, C++ uygulamasını yapınız.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{ setlocale(LC_ALL,"Turkish"); //Türkçe karakter
int adet = 0;
string cumle;
cout<<"Bir Kelime Girin:";
getline(cin,cumle); //kelimeler arasındaki boşlukları da alır cin
//tek başına kullanıldığında almaz.
    for (int i = 0; i < cumle.length(); i++)
    {
        if (cumle[i] == 'a' || cumle[i] == 'e' || cumle[i] == 'ı' ||
cumle[i] == 'i' || cumle[i] == 'u' || cumle[i] == 'ü' || cumle[i] == 'o'
|| cumle[i] == 'ö') adet++;
    }
    cout<<"sesli harf sayısı: "<<adet;
    system("PAUSE"); return 0;}
```

Örnek: Girilen bir cümlede, bulunan kelimeler arasındaki boşluklar olmadan bitişik bir şekilde ekrana yazan konsol uygulamasını yapınız.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{setlocale(LC_ALL,"Turkish"); //Türkçe karakter
int uzunluk;
string metin;
getline(cin,metin);
uzunluk=metin.length();
cout<<"Uzunluk="<<uzunluk<<endl;
for (int a=0;a<=uzunluk-1;a++)
{
    if (metin[a]!= ' ')
    {
        cout<<metin[a];
    }
    else cout<<endl;
}
}
```

Örnek: Kullanıcı tarafından girilen gün adının haftanın kaçınıcı günü olduğunu bulan C uygulaması

```
#include<iostream>
#include <algorithm>
using namespace std;
int main()
{setlocale(LC_ALL,"Turkish"); //Türkçe karakter
string gun;
string gunler[7] = { "pazartesi", "salı", "carsamba", "persembe", "cuma",
"cumartesi", "pazar" };
    cout<<"Gün adı giriniz : ";
    cin>>gun;
    transform(gun.begin(), gun.end(), gun.begin(), ::tolower);
    cout<<gun<<" ";
    for (int i = 0; i <7; i++)
    {
        if (gun == gunler[i])
        {
            cout<<i + 1<<" . gündür";
        }
    }
}
```


Örnek:

```

#include<iostream>
#include<math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int main()
{
    //srand(time(NULL));
    setlocale(LC_ALL,"Turkish"); //Türkçe karakter
    int sayilar[10];
        //Dizi dolduruluyor
    cout<<"Dizi benzersiz sayılarla dolduruluyor..."<<endl;
    for (int i = 0; i < 10; i++)
    {
        sayilar[i] = rand() % 20 + 1;
//oluşturulan sayı dizinin önceki elemanlarında var mı kontrol ediliyor
        for (int kontrol = 0; kontrol < i; kontrol++)
        {
            // Oluşturulan sayı daha önce varsa
            // döngü değişkeni 1 azaltılar sayının tekrar oluşturulması sağlandı
            if (sayilar[kontrol] == sayilar[i])
            {
                i--;
                break;
            }
        }
    }
    //Dizi ekrana yazılıyor
    cout<<"Dizi dolduruldu ekrana yazılıyor..."<<endl;
    for (int a = 0; a < 10; a++)
    {
        cout<<sayilar[a]<<endl;
    }
}

```

Örnek:

```

#include <iostream>
#include<string.h>
using namespace std;
char a[]="Kastamonu";
int main()
{ cout<<"Dizi elemanlari:"<<endl;
  for (int i=0;i<strlen(a);i++)
      cout<<a[i]<<"\n";

  system("PAUSE");
  return 0;
}

```

Örnek:

```
#include <iostream>
#include <string.h>
using namespace std;
char b[50];
int main()
{ strcpy(b,"Ankara");
  for (int i=0;b[i];i++)
    cout<<b[i]<<"\n";

  system("PAUSE");
  return 0;
}
```

Örnek:

```
#include <iostream>
#include <string.h>
using namespace std;
char b[50];
int main()
{ strcpy(b,"Ahmet");
  strcat(b," Yesevi");
  cout<<b<<"\n";
  system("PAUSE");
  return 0;
}
```

Örnek:

```
#include <iostream>
#include <string.h>
using namespace std;
char b[50],c[50];
int main()
{ strcpy(b,"Kazakistan");
  strcpy(c,"Kazak");
  if (strcmp(b,c)==0)
    cout<<"Katarlar esit"<<"\n";
  if (strcmp(b,c)==1)
    cout<<"Birinci katar buyuk"<<"\n";
  if (strcmp(b,c)==-1)
    cout<<"Ikinci katar buyuk"<<"\n";
  system("PAUSE");
  return 0;
}
```

Örnek:

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{ char isim[10];
  cout<<"Cümle Girin=";
  gets(isim);
  int i=0;
  for (i=0;isim[i]!='\0';i++) printf("%c\n",isim[i]);
  cout<<"Uzunluk="<<i<<"\n";
  system("PAUSE");
  return 0;
}
```

KONU SONU ÖDEVLERİ

Ödev: Klavyeden girilen herhangi bir cümlenin içerisinde geçen kelime ve harf sayısını bulan programı yazınız

Ödev: 6-49 Sayısal loto tahmini yapan bir program yazın.

Ödev: Klavyeden girilen 10 sayıdan ortalamaya en yakın olan sayıyı bulan C programını yazın?

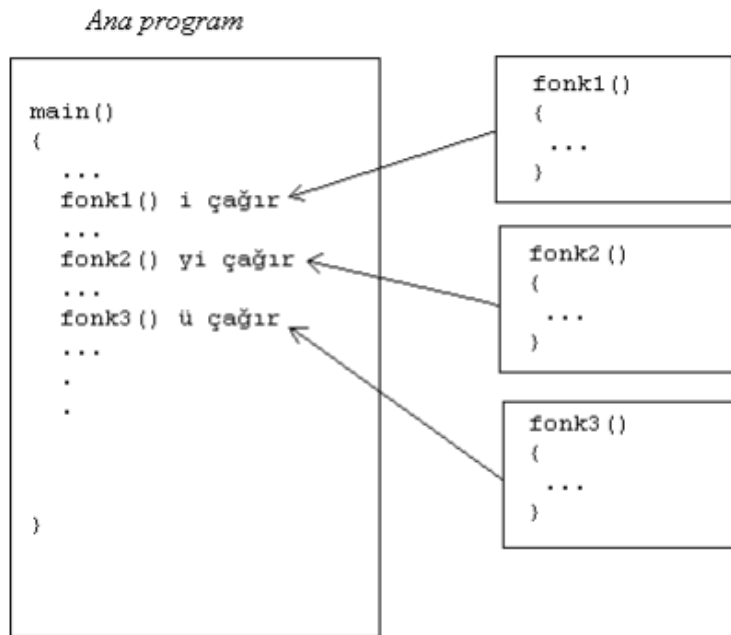
Ödev: Klavyeden girilen bir cümleyi şifreleyen program yazılacaktır. Kullanıcı bir cümle girdikten sonra enter tuşuna bastıktan sonra ilk önce girilen cümle tekrar ekrana yazdırılacak sonra ise cümle tersten şifreli olarak ekrana yazdırılacaktır. Tersten ekrana yazdırılır iken a yerine ?, e yerine *, i veya ı yerine =, ö veya o yerine & ve ü veya u yerine + karakterleri kullanılacaktır. Cümlede kaç tane karakter şifrelendi, kaç tanesi şifrelenmedi bilgisi de ekrana yazdırılacaktır.

Ödev: Takımlar adlı dizi içerisinde 18 tane takım bulunmaktadır(Takım1, Takım2, vb.). Bu takımlar arasında yapılacak olan ilk maçlar için 18 tane takımı eşleştiren bir program yazınız.

FONKSİYONLAR

Fonksiyonlar programlamanın en küçük yapı taşlarıdır. Genel amaçları değişen girdi değerlerine göre sıklıkla hesaplanması gereken işlemlerin gruplanması için kullanılır. Bir işlem ya da işlem grubu program içerisinde birden fazla yerde kullanılıyorsa aynı kodları tekrar, tekrar yazmak yerine fonksiyon tanımlaması yapılabilir.

```
FonksiyonTipi FonksiyonAdı(parametre listesi)
{
Yerel değişkenlerin bildirimi
...
fonksiyon içindeki deyimler veya diğer fonksiyonlar
...
}
```



Fonksiyonları geri dönüş yapan ve yapmayan olarak temelde ikiye ayırabiliriz. Fonksiyonların kullanım yerlerine küçük bir örnek verebiliriz. Örneğin bir program yazıyor olalım ve programda ekrana 10, 20, 50 ve 100 defa yıldız (*) karakterinin yazılması gerekiyor. Bu durumda fonksiyon kullanmadan program şu şekilde olur.

```
int main()
{
for (int i = 0; i < 10; i++) cout<<' * ';
cout<<endl;
for (int i = 0; i < 20; i++) cout<<' * ';
cout<<endl;
for (int i = 0; i < 50; i++) cout<<' * ';
cout<<endl;
for (int i = 0; i < 100; i++) cout<<' * ';
cout<<endl;}
```

Ancak bu işlem fonksiyonla yapılsa aşağıdaki gibi olur.

```
#include<iostream>
using namespace std;
void yildiz_yaz(int kactane)
{
    for (int i = 0; i < kactane; i++) cout<<'*';
    cout<<endl;
}
int main()
{
    yildiz_yaz(10);yildiz_yaz(20);
    yildiz_yaz(50);yildiz_yaz(100);
}
```

Bu programa bakıldığında daha anlaşılır ve sade olduğu görülmektedir. “yildiz_yaz” isimli fonksiyon geriye bir değer döndürmemektedir. Bu durum “void” ile belirtilmektedir. Bir değer döndürmesi istenseydi döndüreceği değerın tipi void yerine yazılmalı ve fonksiyon içinde return komutu kullanılmalıydı.

Aşağıdaki örnekte parametre olarak gönderilen sayının karesini alıp geriye döndüren fonksiyon gösterilmektedir.

```
#include<iostream>

using namespace std;
int kare_al(int);
int main()
{
    cout<<kare_al(10);
}

int kare_al(int sayi)
{
    int kare = sayi * sayi;
    return kare;
}
```

Örnek:

```
#include<iostream>
using namespace std;
int kilodantona_cevir(int sayi)
{
    return sayi/1000;
}

int main()
{
    cout<<kilodantona_cevir(1000);
}
```

Örnek:

```
#include<iostream>
using namespace std;
int kare_al(int);
int ustal(int,int);
int main()
{
    cout<<ustal(5,2)<<endl;
    cout<<ustal(2, 3);
}

int ustal(int taban,int us)
{
    int sonuc = 1;
    for (int i = 0; i < us; i++) sonuc *= taban;
    return sonuc;
}
```

Ödev: Klavyeden girilen sayının asal olup olmadığını geriye döndüren “asalmi” isimli fonksiyonu C’da yazın ve kullanın.

Ödev: Klavyeden girilen sayıyı tersten yazdıran “tersten_yaz” isimli fonksiyonu C’de yazın ve kullanın.

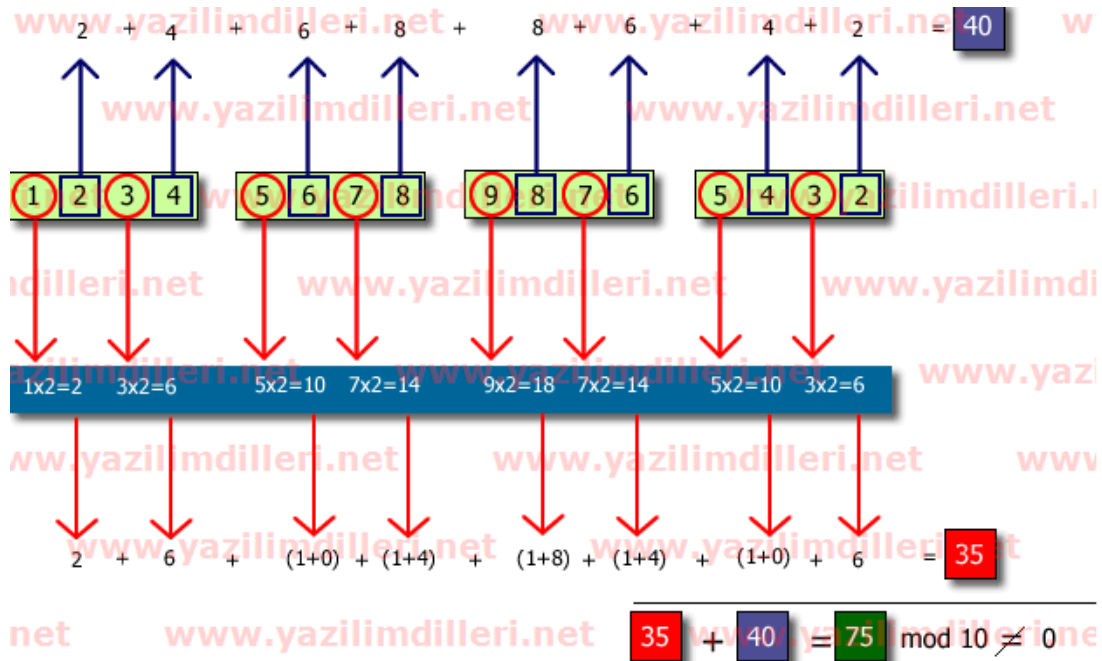
Örnek:

```
#include<iostream>
using namespace std;

int faktoryel(int sayi)
{
    int sonuc = 1;
    for (int i = 1; i <= sayi; i++)
    {
        sonuc *= i;
    }
    return sonuc;
}

int main()
{
    for (int i = 0; i <= 10; i++)
        cout<<i<<"!="<<faktoryel(i)<<endl;
}
```

Kredi Kartı Doğrulama - Luhn Algoritması



- İlk kuralımızın gelen sayı 16 karakter olmak zorunda.
- Bir değişkene indisleri çift olan sayıların iki katlarının basamak toplamlarının, toplamı aktarılmalı.
- Diğer bir değişkene ise indisleri tek olan sayıların toplamı aktarılmalı.
- Son olarak da iki değişkenin toplamının 10 a bölünüp bölünmediğini kontrol edeceğiz.

```
#include<iostream>
using namespace std;
int basamak_topla(int sayi)
{
    int toplam = 0;
    while (sayi != 0)
    {
        toplam += sayi % 10;
        sayi /= 10;
    }
    return toplam;
}

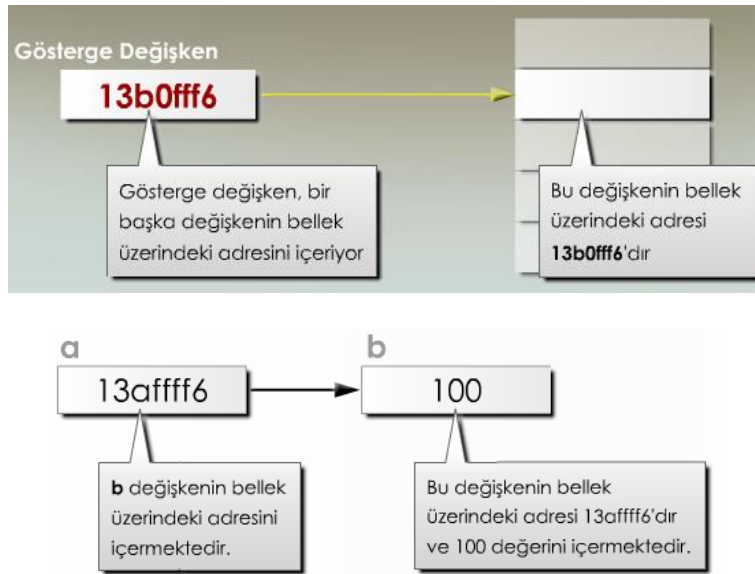
int main()
{
    setlocale(LC_ALL, "Turkish"); //Türkçe karakter
    string kartno;
    kartno = "6472709074972359";
    int tekler=0, ciftler = 0;
    for (int i = 0; i < 16; i++)
    {
        if (i % 2 == 0) ciftler += basamak_topla(2*((int)(kartno[i]-'0')));
        else tekler += ((int)(kartno[i]-'0'));
    }

    if ((tekler+ciftler)%10==0) cout<<"Doğru Kart Numarası"<<endl;
    else cout<<"Yanlış Kart Numarası"<<endl;
}
```

Ödev: Luhn algoritmasına göre olası bütün doğru kredi kartı numaralarını bulup listeleyen C programını yazın.

POINTER (Göstergeç) Kavramı

Gösterge (*pointer*) C programlarında önemli yere sahip bir kavramdır. Gösterge, bir değişkenin bellekteki adresini tutan bir değişken olarak düşünülür. Örneğin, b değişkeninin bellekteki konumunu, yani adresini a değişkeni içine yerleştirelim. Bu durumda "a, b'nin göstergesidir" ya da "a, b'ye işaret etmektedir" denir.



Göstergelerin Bildirimi

tür * değişken adı

- Bildirimi yapılmış bir gösterge değişken herhangi bir şey ifade etmez.
- Bu göstergenin kullanılabilmesi için, söz konusu gösterge değişkene bir başka değişkenin adresini yerleştirmek gerekmektedir.
- Bir değişkenin bellek üzerindeki adresini öğrenmek için "&" işlecinden yararlanılır.

Örnek:

```
#include<iostream>
using namespace std;
int main()
{setlocale(LC_ALL,"Turkish"); //Türkçe karakter
  int a = 5;
  int* p = &a;
  cout<<a<<endl;
  cout<<&a<<endl;
  cout<<p<<endl;
}
```


Örnek:

```
#include<iostream>
using namespace std;
int main()
{
    int yas;
    int* p;
    yas = 5;
    p = &yas;
    cout<<*p<<endl;
    cout<<p<<&yas<<endl;
    *p+= 3;
    cout<<yas<<endl;
}
```

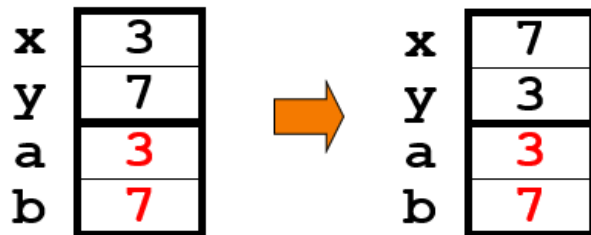
- C programlama dilinde; tür *değişken adı şeklinde tanımlanır.
- Örnek: int *a; //tamsayı türünden bir pointer tanımlı.
- *: Veri tipinin hemen sağına ya da değişken adının soluna yazılarak, belirtilen tipteki değişken için pointer tanımlar.
- *: Tanımlanmış herhangi bir pointer'ın önüne yazıldığı zaman ise, o pointer'ın tuttuğu adresteki içeriği döndürür.
- & :Daha önce tanımlanmış herhangi tipteki değişkenin bellek adresini döndürür. Bu adres ancak aynı tipte tanımlanmış bir pointer değişkene atanabilir.

Örnek: Değerle Çağrı Yapmak

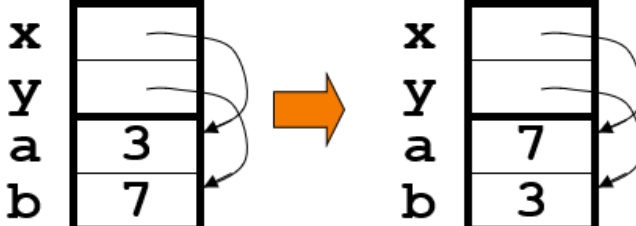
```
void swap(int x, int y)
{
    int t;
    t = x;
    x = y;
    y = t;
}

int main(void) {
    ...
    swap(a,b);
    ...
}
```

Fonksiyon parametreleri değerle çağrılır.
Değerler “yerel değişkenlere” kopyalanır.



Örnek: Referansla Çağrı Yapmak

<pre> void swap(int *x, int *y) { int t; t = *x; *x = *y; *y = t; } int main(void) { ... swap(&a,&b); ... } </pre>	<p>‘Referans’ ile değerleri aktarmak için göstericiler kullanılır.</p> 
---	---

Pointer Aritmetiği

- Pointerlar üzerinde yapılan aritmetik toplama ve çıkarma işlemlerinde artış değeri, pointer'ın sahip olduğu veri tipinin bellekte kapladığı alan kadar olmaktadır.
- Böylece ardışık bellek alanlarında bulunan aynı tipteki verilere (dizi) erişim kolaylıkla mümkün hale gelir.
- Örneğin,
char *p; p++; // p değeri 1 byte artar
int *p; p++; // p değeri standart tamsayı için 4 byte artar

Örnek:

```

#include<iostream>
using namespace std;
int main()
{
    int a = 500; int* b;
    b = &a;
    cout<<"Adres:"<<b<<endl;
    cout<<"Sonraki:"<<(b + 1)<<endl;
    cout<<"Önceki:"<<(b - 1)<<endl;
    cout<<endl;
    cout<<"Adres:"<<&a<<endl;
    cout<<"Sonraki:"<<(&a + 1)<<endl;
    cout<<"Önceki:"<<(&a - 1)<<endl;
}

```

```

Adres:0x70fe04
Sonraki:0x70fe08
Önceki:0x70fe00

Adres:0x70fe04
Sonraki:0x70fe08
Önceki:0x70fe00

```

Örnek:

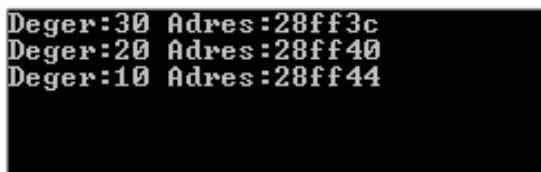
```
#include<stdio.h>
#include<conio.h>
main()
{
    int a=10,b=20,c=30; int *p;
    p=&c;
    *(p+2)=a+b;
    printf("a:%d \n",a);
    printf("b:%d \n",b);
    printf("c:%d \n",c);
    getch();
}
```



```
a:10
b:20
c:30
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{
    int a=10, b=20, c=30; int *p;
    p=&c;
    printf("Deger:%d Adres:%x\n",*p,p);p++;
    printf("Deger:%d Adres:%x\n",*p,p);p++;
    printf("Deger:%d Adres:%x\n",*p,p);
    getch();
}
```



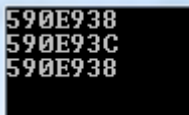
```
Deger:30 Adres:28ff3c
Deger:20 Adres:28ff40
Deger:10 Adres:28ff44
```

Örnek:

```
#include <iostream>
using namespace std;
main()
{
    int x;
    int* p;
    p = &x; *p = 1200;
    cout<< *p<<endl;
    cout<< --(*p)<<endl;
    cout<< ++(*p)<<endl;
}
```

Örnek:

```
#include <iostream>
using namespace std;
main()
{
    int x;
    int* p;
    p = &x; *p = 1200;
    cout<<p<<endl;
    cout<<++p<<endl;
    cout<<--p<<endl;
}
```



```
590E938
590E93C
590E938
```

Örnek:

```
int main()
{
    int a = 10; int* p;
    p = &a;
    cout<<*&a<<endl; }
```

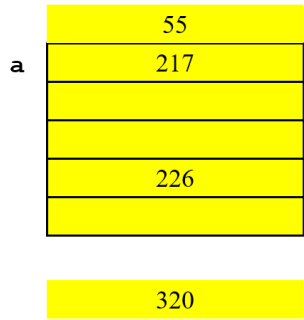
Dizilerle Pointer Kullanımı

- Bir a[] dizisinin temel adresi &a[0] biçiminde elde edilir.
- Aslında bir dizi indissiz olarak kullanılırsa, bu dizi adı da dizinin temel adresini verir. Örneğin, a[] dizisinin temel adresini bulmak için sadece a yazmak da yeterlidir.

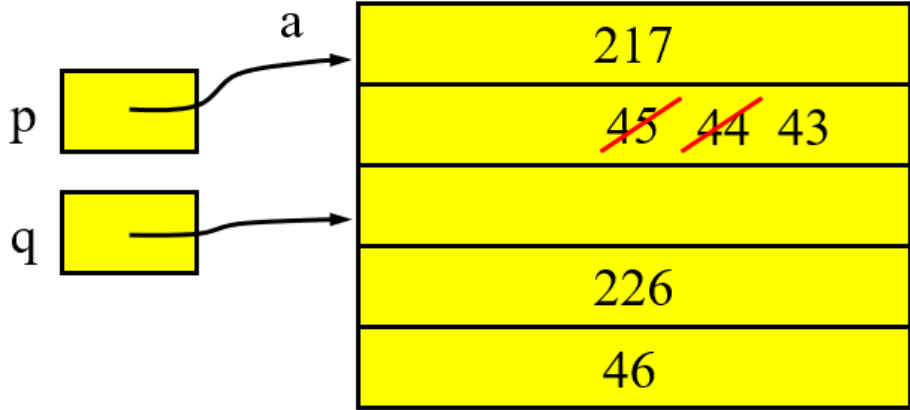


- Bir dizinin ardışık elemanları bellekte de ardışık bir şekilde bulunur.
- Bu nedenle çoğu zaman herhangi dizi, pointerlar ile kullanmaya uygundur.

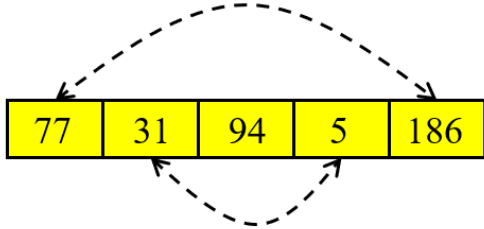
Örnek: Aşağıdaki kod parçasının çalışmasıyla bellekte nasıl bir hareket olduğuna bakalım:

<pre>... int a[5]; a[0] = 217; a[3] = 226; ...</pre>	
<pre>... a[-1] = 55; a[7] = 320; ...</pre>	<p>a dizisinden önce veya sonra yer alan <u>başka</u> bir değişkene ait bellek bölgesini işgal etti! Çünkü, C sınırları kontrol etmez!</p>

Örnek: Aşağıdaki kod parçasının çalışmasıyla bellekte nasıl bir hareket olduğuna bakalım:

<pre>... int a[5]; int *p, *q; p = a; p[1]= 44; q = p + 2; q[-1] = 43; q[2] = 46; ...</pre>	
--	---

Örnek: Diziyi tersleme fonksiyonunu yazalım:

<p>Dizideki değerleri ters çevir</p> <p>Giriş: a tamsayı dizisi, n eleman sayısı</p> <p>Çıkış: a'nın değerlerinin ters sırada yer aldığı dizi</p> <p>Algoritma</p> <p>Dizideki ilk ve son elemanı yer değiştir</p> <p>İkinciyle sondan ikinciye yer değiştir</p> <p>...</p>	
<pre>void ters_cevir (int a[], int n) { int l, r, temp; for (l=0, r=n-1; l<r; l++, r--) { temp = a[l]; a[l] = a[r]; a[r] = temp; } }</pre>	

```
int main(void) {
    ...
    ters_cevir (dizi, 5);
    ...
}
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{
    int puan[5]; int *p;
    p = puan; //p = &puan[0];
    puan[3] = 50; // ifadesi ile aşağıdaki ifade aynıdır;
    *(p+3) = 55; // değeri pointer yoluyla aktarma
    printf("puan[3]:%d \n",*(p+3));
    getch(); }
```

Örnek:

<pre>#include <stdio.h> #include <conio.h> main() { char *k="Ahmet Yesevi"; printf(k); getch(); }</pre>	<pre>#include <stdio.h> #include <conio.h> main() { char a[]="Ahmet Yesevi"; char *p=a; printf(a); printf(p); getch(); }</pre>	<pre>#include <stdio.h> #include <conio.h> main() { char a[]="Ahmet Yesevi"; printf(a); getch(); }</pre>
---	--	--

Ahmet YeseviAhmet Yesevi

```
#include<stdio.h>
#include<conio.h>
main()
{
    char *isim="Merhaba";int uzunluk;
    puts(isim);
    for (;*isim!='\0';isim++)
        printf("%c \n",*isim);
    getch();
}
```



```
Merhaba
M
e
r
h
a
b
a
```

```
#include<stdio.h>
#include<conio.h>
main()
{   char *a[10];int uzunluk;
    a[0]="Ekle"; a[1]="Sil";
    printf("%s \n",a[0]);
    printf("%s \n",a[1]);
    uzunluk=0;
    for (int i=0;*(a[0]+i);i++)
        { printf("%c\n",*(a[0]+i));
          uzunluk++; }
    printf("%s kelimesinin uzunluđu=%d\n",a[0],uzunluk);
    getch();
}
```

```
Ekle
Sil
E
k
l
e
Ekle kelimesinin uzunlu-u=4
```

Örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{   char *p[]={ "Samsun","Ordu","Ankara","Sinop" };
    char a[][10]={ "Samsun","Ordu","Ankara","Sinop" };
    for (int i=0;i<4;i++) printf ("%s\n",p[i]); //%s adres ister
    for (int i=0;i<4;i++) printf ("%s\n",a[i]);
    for (int i=0;i<4;i++) printf ("%s\n",&a[i][0]);
    getch();
}
```

```
Samsun
Ordu
Ankara
Sinop
Samsun
Ordu
Ankara
Sinop
Samsun
Ordu
Ankara
Sinop
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
main()
{
    char *isim="Merhaba";int uzunluk;
    puts(isim);
    for (;*isim!='\0';isim++)
        printf("%c \n",*isim);
    getch();
}
```

Örnek: Tam Sayı Listesini Sıralayan Fonksiyon

- Problem
 - Girdi: n adet v tamsayı dizisi
 - Çıktı: küçükten büyüğe sıralı dizi
- Sıralamanın tonlarca yolu var, fakat ortak şu üç özelliğe sahipler
 - İki eleman arasında karşılaştırma
 - İki elemanın sırasını ters çevirme
 - Tamamlanıncaya değin karşılaştır ve yerdeğıştir (takas)

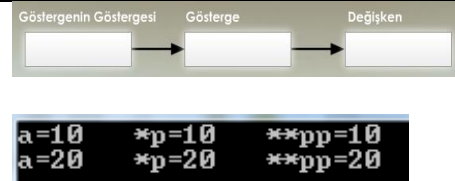
- En basit yaklaşım
 - Soldan sağa ilerle
 - Bire bir kıyasla, daha küçük olanı soldaki yerdegiş

```
void sort(int *v, int n)
{
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = i+1; j < n; j++) {
            if (v[i] > v[j]) { //karşılaştırma işlemi
                int swap = v[i]; // yer değıştirme(takas) işlemi
                v[i] = v[j];
                v[j] = swap;
            }
        }
    }
}
```

Pointer'ı İşaret Eden Pointer

- Bir göstergeye bir başka göstergenin işaret etmesi mümkündür.
- Bu durumdaki göstergelerin başında ** işleci yer alır. Örneğin, **a göstergesi bu tür bir göstergedir.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,*p,**pp;
    p=&a; pp=&p; a=10;
    printf("a=%d *p=%d **pp=%d\n",a, *p,**pp);
    **pp=20;
    printf("a=%d *p=%d
    **pp=%d",a, *p,**pp);
    getch();
}
```



Karakter Dizisi – Dizgi

Harf dizinleri metin adıyla bildiğimiz türdür. C dilinde string'ler asıl veri türlerinden birisi değildir.

Char türünden bir boyutlu diziler olarak elde edilir.

printf() ve scanf() fonksiyonlarının argümanları olarak çok kullanılırlar.

C dilinin temel veri türlerinden olmadığı için, string'lerin kullanılış biçimi dizilerin kullanılışından farklı değildir; yani bir string'in öğelerine ancak tek tek erişmek olanağı vardır.

C dilinde string'ler aşağıdaki yöntemlerle kurulabilir:

1. Yöntem

İstenen metin " " simgeleri arasına yazılır. Örneğin,

```
printf("C Programming");
```

deyiminde " " işaretleri arasına alınan metin bir harf dizini (metin,string) oluşturur. Buradaki dizin printf() fonksiyonunun argümanı (bağımsız değişkeni) rolünü oynar. " " içindeki bir metnin son harfinden sonra, C derleyicisi metnin sonuna gelindiğini belirtmek üzere \0 karakterini koyar. Standart cıkista (ekran, printer) görünmeyen bu karaktere

NULL terminatör adı verilir ve enter tuşuna basıldığında derleyici tarafından kendiliğinden oluşturulur.

2. Yöntem

İstenen metni içerebilecek bir yeri ana bellekte ayırmak için, yeterli büyüklükte bir dizi yaratılır. Örneğin,

```
char metin[20] ;
```

bildirimini 20 öğeli bir dizi yaratır.

Her bir öge bir harf temsil edecek bir değişkendir. Bu değişkene değer atamak için

```
metin[] = "Merhaba" ;
```

```
metin[] = {'M', 'e', 'r', 'h', 'a', 'b', 'a', '\0'};
```

Burada '\0' simgesi NULL (bos) karakter adıyla anılan değerdir. String'in bittiğini belirtir.

3. Yöntem

String bildiriminde pointerlerden yararlanabiliriz. Örneğin,

```
char *soyad ;
```

bildiriminden sonra

```
soyad = "Deniz";
```

ataması geçerlidir. Bu adamada 'D' harfi soyad pointerinin gösterdiği adrese yerleşir. Öteki harfler , o adresten sonraki 1 er byte'lık bellek hücrelerine sırayla yerleşirler. Dolayısıyla,

```
*soyad == 'D'
*(soyad +1) == 'e'
*(soyad +2) == 'n'
*(soyad +3) == 'i'
*(soyad +4) == 'z'
*(soyad +5) == '\0'
```

Örnek: Dizgileri Sıralama

- İpucu: Karşılaştırma işleci farklıdır
 - Büyük mü işleci (">") işe yaramaz
 - strcmp()** kullanmak zorundayız. (string kıyaslama metodu)

```
void sort(char *v[], int n)
{
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = i+1; j < n; j++) {
            if (strcmp(v[i], v[j]) > 0) {
                char* swap = v[i];
                v[i] = v[j];
                v[j] = swap;
            }
        }
    }
}
```

Çalışma Soruları

1-) Kendisine verilen iki sayının OKEK (Ortak Katların En Küçüğü) değerini hesaplayıp, geriye döndüren fonksiyonu yazınız

2-)Kendisine verilen iki sayının OBEB (Ortak Bölenlerin En Büyüğü) değerini hesaplayıp, geriye döndüren fonksiyonu yazınız.

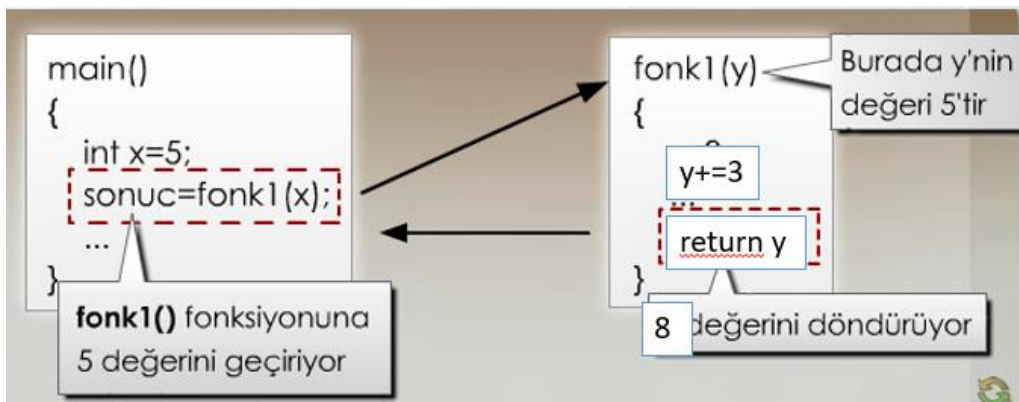
- 3-) En ve boy parametrelerine göre, '*' simgeleriyle dikdörtgen çizen bir fonksiyon yazınız?
- 4-) Kendisine argüman olarak verilen bir tamsayıyı tersine çevirip, sonucu döndürecek bir fonksiyon yazınız?
- 5-) Bir tamsayının faktöriyelini hesaplayıp, geriye döndüren fonksiyonu yazınız.?
- 6-) Vize final notunu girerek ortalama ve harf notunu yazdırınız, (fonksiyon kullanınız)?
(8 tane harf :FF(0-30),DD(30-40),DC(40-50),CC(50-60),CB(60-70),BB(70-80),BA(80-90),AA(90-100))
- 7-) N1 –N2 e kadar olan sayılardan 3 e bölünemeyenleri bulup ekrana yazdırınız? (fonksiyon kullanınız)

8-) Aşağıdaki programın bellek değişkenlerinin değişimini gösterin.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=5;int *y;
    y=&x;
    --*y--;
    *++y=15;
    getch();
}
```

Değer ile Fonksiyon Çağırma

- Bu çağırma biçiminde; çağıran fonksiyondan çağrılan fonksiyona aktarılan değişken değerlerinde bir değişiklik olsa bile, bu değişiklik çağıran programdaki değişken değerlerine etki etmez.
- Bu değişiklikler çağrılan fonksiyon içinde parametrelerin kopyaları üzerinde gerçekleşir.



Örnek:

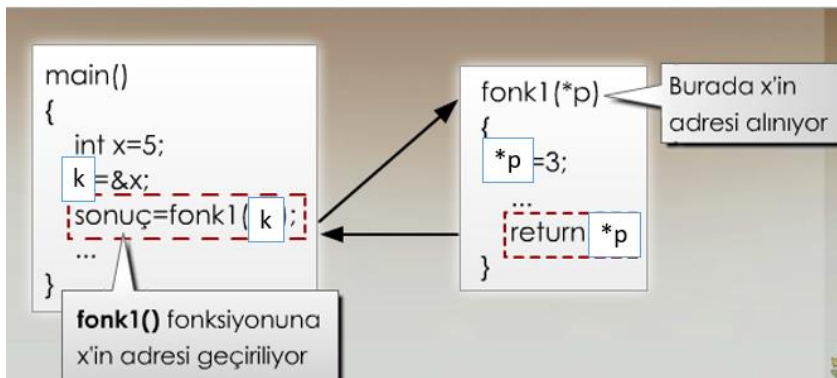
```
#include <stdio.h>
#include <conio.h>
int artir(int);
main()
{
    int deger=10;
    printf("Deger=%d\n",deger);

    printf("Deger=%d\n",artir(deger));
    printf("Deger=%d\n",deger);
    getch();
}
int artir(int tutdeger)
{
    tutdeger+=5;
    return tutdeger;
}
```

```
Deger=10
Deger=15
Deger=10
```

Referans ile Fonksiyon Çağırma

- Verilerin bellek adreslerini kullanarak da fonksiyonları çağırabiliriz.
- Bu durumda, çağrılan fonksiyon içinde parametre değerleri üzerinde yapılacak değişikliklerin, onu çağırın fonksiyon üzerinde de etkili olduğu görülecektir.

**Örnek:**

```
#include <stdio.h>
#include <conio.h>
int artir(int*);
main()
{
    int deger=10;
    printf("Deger=%d\n",deger);

    printf("Deger=%d\n",artir(&deger));
    printf("Deger=%d\n",deger);
    getch();
}
int artir(int *tutdeger)
{
    *tutdeger+=5;
    return *tutdeger;
}
```

```
Deger=10
Deger=15
Deger=15
```

Örnek:

```
#include <iostream>
using namespace std;
int referansilecagirma(int &sayiref)
{ sayiref*=sayiref;
}
int main()
{ int y=4;
  cout<<"Fonksiyonu cagirmadan once y="<<y<<endl;
  referansilecagirma(y);
  cout<<"Fonksiyonu cagirdiktan sonra y="<<y<<endl;
  system("PAUSE");
  return 0;
}
```

```
Fonksiyonu cagirmadan once y=4
Fonksiyonu cagirdiktan sonra y=16
Devam etmek için bir tuşa basın . . .
```

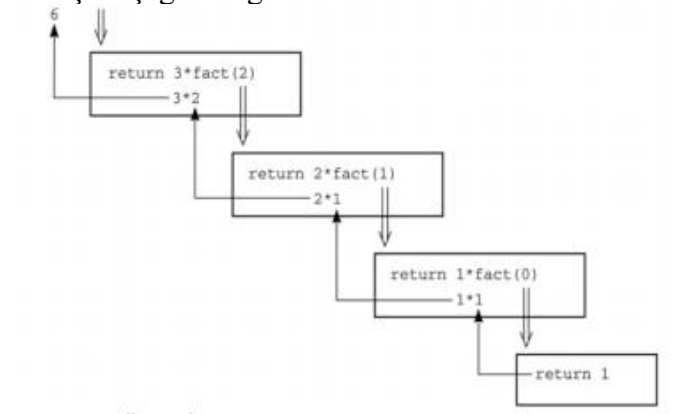
Recursive (Özyineleme)

- Kendi kendisini doğrudan veya dolaylı olarak çağıran fonksiyonlara özyineli (recursive) fonksiyonlar adı verilir.
- Özyineleme bir problemi benzer şekilde olan daha küçük parçalara bölünerek çözülmesini sağlayan bir tekniktir.
- Özyineleme, döngülere (iteration) alternatif olarak kullanılabilir.

```
#include <stdio.h>
#include <conio.h>
int fakt(int);
main()
{ printf("%d",fakt(5));
  getch();
}
int fakt(int n)
{ if (n <= 1) return 1;
  else return n * fakt(n - 1);
}
```

5*fakt(4) (5*24=120)
 4*fakt(3) (4*6=24)
 3*fakt(2) (3*2=6)
 2*fakt(1) (2*1=2)

n=3 için aşağıdaki gibi olur.



Örnek : Fibonacci dizisinin özyineleme ile bulunması:

Fibonacci dizisinde her eleman kendinden önceki iki elemanın toplamı şeklinde hesaplanır.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

0.eleman : 0

1.eleman : 1

2.eleman : $0+1 = 1$

3.eleman : $1+1 = 2$

4.eleman : $1+2 = 3$

5.eleman : $2+3 = 5$

Fibonacci dizisinin tanımı:

$\text{fib}(n) = n$ if $((n==0) \parallel (n==1))$

$\text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1)$ if $(n \geq 2)$

Fibonacci dizisi program:

```
int fib(int n)
{   if (n<=1) return n;
    else return fib(n-1)+fib(n-2);
}
```

Örnek hesaplama :

$\text{fib}(4) = \text{fib}(2) + \text{fib}(3)$
 $= \text{fib}(0) + \text{fib}(1) + \text{fib}(1) + \text{fib}(2)$
 $= 0 + 1 + 1 + \text{fib}(0) + \text{fib}(1)$
 $= 2 + 0 + 1$
 $= 3$

Örnek: Aşağıdaki Programı izleyin değişken değişimlerini gösterin. Programın yaptığı işi bir cümle ile belirtin.

```
int main()
{   cout<<topla(4)<<endl; }

static int topla(int n)
{
    if (n <= 1) return n;
    else return n + topla(n - 1); }
```

STRUCTS(YAPILAR)

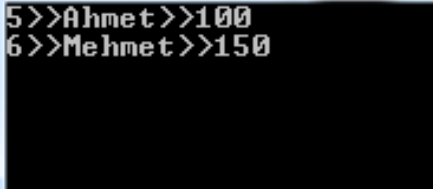
- C programlama dilinde birbirleriyle ilişkili veri gruplarına yapı (struct) adı verilmektedir.
- Yapı, farklı veri türlerine sahip değişkenlerin bir grup olarak değerlendirilmesi ve bu grubun bir isimle kullanılması amacıyla tercih edilir.
- Yapılar aşağıdaki gibi tanımlanıyor:

Struct yapı-adı { alanlar } değişken listesi;	Farklı değişkenler gruplandırılarak bir yapı elde edilir 
---	---

- Farklı şekilde tanımlanabilir. En basit tanımlama;
struct yapı
 {
 int x;
 char y[5];
 int z;
 } **test;**
- Bu yapının her bir alanına C programının herhangi bir yerinde erişilebilir.
- Bunun için, yapı değişkeni ve alanı birbirinden (.) işleci ile ayrılmak suretiyle kullanılır.
 Yapının x alanını,
 test.x=5; printf("%d", test.x); şeklinde kullanabiliriz.

<pre>struct liste{ char *adsoyad; float vize; float final; } ogrenciler; main() { ogrenciler.adsoyad="Ahmet"; ogrenciler.vize=50; ogrenciler.final=100; }</pre>	<pre>struct liste{ char *adsoyad; float vize; float final; }; struct liste ogrenciler; main() { ogrenciler.adsoyad="Ahmet"; ogrenciler.vize=50; ogrenciler.final=100; }</pre>	<pre>typedef struct listetipi{ char *adsoyad; float vize; float final; } liste; liste ogrenciler; main() { ogrenciler.adsoyad="Ahmet"; ogrenciler.vize=50; ogrenciler.final=100; }</pre>
---	---	--

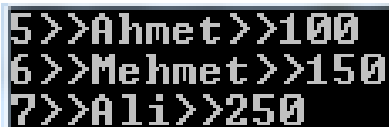
Birden Fazla Yapı Değişkeninin Kullanımı

<pre>#include<stdio.h> #include<string.h> typedef struct yapi { int numara; char adi[10]; int ucret;} personel; personel p1,p2; main() { p1.numara=5; strcpy(p1.adi,"Ahmet"); p1.ucret=100; printf("%d>>%s>>%d\n",p1.numara,p1.adi, p1.ucret); p2.numara=6; strcpy(p2.adi,"Mehmet"); p2.ucret=150; printf("%d>>%s>>%d",p2.numara,p2.adi, p2.ucret); getch(); }</pre>	
--	--

Yapı Dizileri

- Diziler yardımıyla, aynı tür veriler bir dizi biçiminde bellekte yerleştirilmekte ve bir indeks yardımıyla bu dizi elemanlarına erişilebilmektedir.

<pre>#include<stdio.h> #include<conio.h> #include<string.h> typedef struct yapi { int numara; char adi[10]; int ucret; } personel; personel p[3];</pre>	<pre>main() { p[0].numara=5; strcpy(p[0].adi,"Ahmet"); p[0].ucret=100; p[1].numara=6; strcpy(p[1].adi,"Mehmet"); p[1].ucret=150; p[2].numara=7; strcpy(p[2].adi,"Ali"); p[2].ucret=250; for(int i=0;i<3;i++) printf("%d>>%s>>%d\n",p[i].numara,p[i].adi, p[i].ucret); getch(); }</pre>
---	--



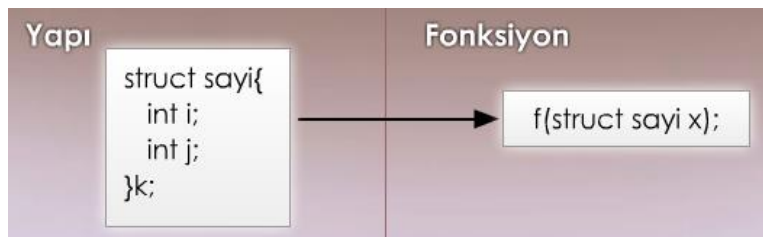
Alanlara Başlangıç Değeri Vermek

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct listetipi{
    char *adsoyad;
    float vize;
    float final;
} liste;
liste ogrenciler={"Ali",30,60};
main()
{ printf("%s\n",ogrenciler.adsoyad);
  printf("%3.1f\n",ogrenciler.vize);
  printf("%3.1f\n",ogrenciler.final);
  getch();}
```

```
Ali
30.0
60.0
```

Fonksiyonlarla Kullanımı

- Bir yapı bir fonksiyona, aynen diğer veri türlerinde olduğu gibi parametre biçiminde geçirilebilir.
- Ayrıca bir fonksiyon bir yapıyı kendisini çağıran fonksiyona döndürebilir.
-



```
#include<stdio.h>
#include<conio.h>
#include<string.h>
typedef struct listetipi{
    char *adsoyad;
    float vize;
    float final;
} liste;
void yaz(liste);
main()
{   liste ogrenciler={"Ali",30,60};
    yaz(ogrenciler);

    getch();}
```

```
void yaz(liste yazilacak)
{ printf("%s\n", yazilacak.adsoyad);
  printf("%3.1f\n",yazilacak.vize);
  printf("%3.1f\n",yazilacak.final);
}
```



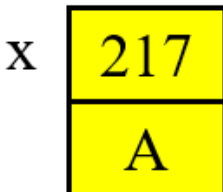
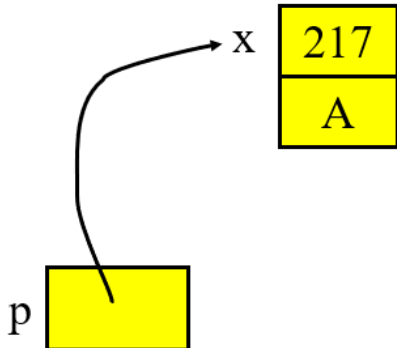
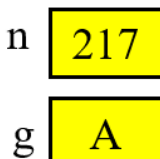
```

Ali
30.0
60.0

```

<pre> #include<stdio.h> #include<conio.h> #include<string.h> typedef struct listetipi{ char *adsoyad; float vize; float final; } liste; void yaz(liste *,int); main() { liste ogrenciler[2]={"Ali",30,60,"Veli",49,50}; yaz(ogrenciler,0); getch(); } </pre>	<pre> void yaz(liste *yazilacak,int i) {printf("%s\n", (yazilacak+i)->adsoyad); printf("%3.1f\n", (yazilacak+i)->vize); printf("%3.1f\n", (yazilacak+i)->final); (yazilacak+i)->vize=100; //ne değışti } </pre>
--	--

Yapı Göstericiler

<pre> struct pair { int number; char grade; }; struct pair x; x.number=217; x.grade='A'; </pre>	 <p>A yellow box divided into two horizontal sections. The top section contains the number 217 and the bottom section contains the letter A. To the left of the box is the label 'x'.</p>
<pre> struct pair *p; p = &x; </pre>	 <p>A yellow box divided into two horizontal sections containing 217 and A, with the label 'x' to its right. Below it is another yellow box with the label 'p' to its left. A curved arrow points from the 'p' box to the 'x' box.</p>
<pre> int n = (*p).number; veya int n = p->number; char g = (*p).grade; veya char g = p->grade; </pre>	 <p>Two yellow boxes are shown. The top box contains 217 and is labeled 'n' to its left. The bottom box contains A and is labeled 'g' to its left.</p>

Birlikler (Union)

- Birlikler yapılar gibi birden fazla alana sahip olan ve her bir alan için farklı veri türü atayabilen yapılardır.
- Ancak, birliklerde yapılar gibi her alana hafızada ayrı yer ayrılmaz bunun yerine hepsi için ortak bir yer ayrılır.
- En son değer atanan hangisi ise hafızadaki alanı kullanır diğerlerinin ise değeri yoktur.
- Dolayısıyla, hafızadaki ayrılmış bir alanı hepsi ortak kullanır en son değer atanan hangisi ise ona tahsis edilir.
- Alanlardan hangisi en fazla hafıza alanına ihtiyaç duyarsa onun gereksinim duyduğu kadar yer ayrılır.
- Yapılara benzeyen bir diğer kavram, birlik adıyla bilinir. Yapılar, belirli bir grup alanın tek bir değişken biçiminde ifade edilmesini sağlıyordu. Değişkenin her bir alanı için bellekte ayrı bir yer ayrılıyordu. Dolayısıyla her bir alan gerek duyulduğunda tek tek kullanılabilirdi.
- Birliklerde ise, birliğin tümü için bellekte tek bir yer ayrılır. Bu yer, birliğin en büyük alanı kadardır. Birlikler aşağıdaki şekilde tanımlanmaktadır

```
union birlik_adı
{
    alanlar
} değişken_listesi;
```



- Ortaklık, birden çok değişkenin aynı bellek alanını kullanmasını sağlayan veri yapısı tanımlamasıdır.
- Böylece, bellek alanı kısıtlı uygulamalarda bazı değişkenlerin ortaklaşa yer kullanılması sağlanmış olunur.
- Ancak, birine atama yapıldığında diğerinin değişeceği unutulmamalıdır.

```
union birlik {
    char h;
    int t;
    float f;
}
union birlik x;
```

- Programcı, hangi anda hangi bileşen değerinin x 'in yerinde olduğunu bilmek zorundadır. Bunu bilmezse, program ciddi yanlışlar yapabilir.

f (4 byte)			
		t (2 byte)	
			h
3. sekizli	2. sekizli	1. sekizli	0. sekizli

Örnek:

```
#include <iostream>
using namespace std;
union birlik {
    char h;    int t;    float f; };
union birlik x;
int main()
{ x.h='a';
  cout<<"h="<<x.h<<"\n";
  x.t=5;
  cout<<"x="<<x.t<<" h="<<x.h<<"\n"; //h için 5'in Ascii kod karşılığı
yazılır
  x.f=10.5;
  cout<<"t="<<x.t<<" h="<<x.h<<" f="<<x.f<<"\n";
  system("PAUSE");
  return 0;
}
```

```
h=a
x=5 h=5
t=1093140480 h= f=10.5
Devam etmek için bir tuşa
```

Örnek:

```
#include <iostream>
using namespace std;
union birlik { char t; int h; };
union birlik x;
main()
{ x.h='A';
  cout<<x.h<<"<";
  x.t=65;
  cout<<x.t++;
  cout<<">"<<(char)x.h;
  cout<<">"<<--x.h;
  system("PAUSE");
  return 0;
}
```

```
65<A>B>65Dev
```

C++ ile Dosya İşlemleri

- Disk dosyalarıyla çalışmak için C++'ın bazı sınıflarını kullanmak gerekiyor.
- Girdi işlemleri için ifstream; çıktı işlemleri için ofstream ve hem girdi hem de çıktı işlemleri için fstream sınıflarına başvurulur.
- Bu sınıfları kullanabilmek için fstream kütüphanesi import edilmelidir.
- Eğer bir programda dosyanın veri kaydetmek ve veri okumak için ayrı ayrı iki kez açılması gerekiyorsa, birinci işlemin tamamlanması ardından yeniden açılabilmesi için kapatılması gerekecektir.

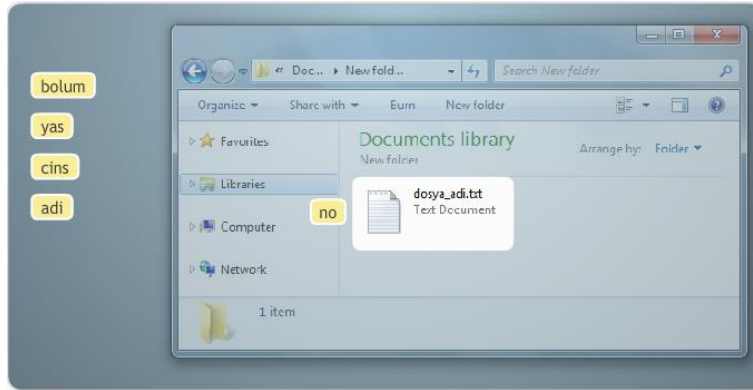
Disk üzerine verileri yazmak için ofstream nesnesi şu şekilde açılmaktadır:

ofstream akım adı("dosya_adı");

Dosya üzerine veri kaydetmek için doğrudan akım nesnesi kullanılır. Örneğin, akım adı dosya ise, bu akımın açılması ve bir değerlerin kaydedilmesi, "<<" işleci kullanılarak şu şekilde olacaktır:

dosya << no << ' ' << adi << ' ' << cins << ' ' << yas << ' ' << bolum;

dosyaya, no, adi, cins, yas ve bölüm değişkenlerinin aralarına birer boşluk yerleştirerek kaydedilmesini sağlar. Bu şekilde yapılan kayıt işlemleri sonucunda, sayısal değerler dahil tüm değişken değerleri disk üzerine karakter olarak kaydedilir.



Örnek: Text Dosyaya Yazma

```
using namespace std;
struct Ogrenci
{int no;string adi;
  char cins;
  int yas;
  string bolum;
};

int main()
{ //çıkış dosyası tanımlanıyor
  ofstream dosya("personel.txt",ios::app); //her seferinde dosya içindeki
                                           //bilgiler silinerek açılır
  // ofstream dosya("personel.txt",ios::app); //ekleme modunda açar
  Ogrenci ogr;
```

```

ogr.no=123;
ogr.adi="BURAK";
ogr.cins='E';
ogr.yas=25;
ogr.bolum="BilgisayarMüh.";
dosya<<ogr.no<<"\n"<<ogr.adi<<"\n"<<ogr.cins
    <<"\n"<<ogr.yas<<"\n"<<ogr.bolum<<"\n";
dosya.close();
cout<<"Bilgiler kaydedildi...\n";
system("PAUSE");
return 0;
}

```

Örnek: Text Dosyadan Okuma

```

#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct Ogresci
{int no;string adi;
  char cins;
  int yas;
  string bolum;
};

int main()
{ //giriş yapılacak dosyası tanımlanıyor
  ifstream dosya("personel.txt");
  for (int i=1;i<=2;i++)
  { Ogresci ogr;
    dosya>>ogr.no>>ogr.adi>>ogr.cins >>ogr.yas>>ogr.bolum;
    cout<<"Numara\t: "<<ogr.no
      <<"\nAd \t: "<<ogr.adi
      <<"\nCins. \t: "<<ogr.cins
      <<"\nYas \t: "<<ogr.yas
      <<"\nBolum \t: "<<ogr.bolum<<endl;
  }
  dosya.close();
  cout<<"Bilgiler okundu...\n";
  system("PAUSE");
  return 0;
}

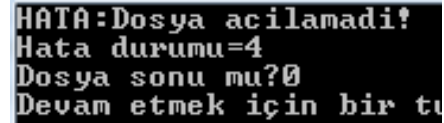
```

Dosya İşlemlerindeki Hata Hakkında Bilgi Almak

Giriş/çıkış durum bilgisini elde etmek için `rdstate()` fonksiyonunun kullanılır.

Örnek:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct Ogrenci
{int no;string adi;
char cins;
int yas;
string bolum;
};
```



```
HATA:Dosya acilamadi!
Hata durumu=4
Dosya sonu mu?0
Devam etmek için bir tu
```

```
int main()
{   Ogrenci ogr;
    ifstream dosyaoku("personel1.txt");
    if (!dosyaoku)
    {cout<<"HATA:Dosya acilamadi!"<<endl;
      cout<<"Hata durumu="<<dosyaoku.rdstate()<<endl;
      cout<<"Dosya sonu mu?"<<dosyaoku.eof()<<endl;
      system("pause");
      return 1;
    }

    dosyaoku>>ogr.no>>ogr.adi>>ogr.cins>>ogr.yas>>ogr.bolum;
    cout<<"Numara\t: "<<ogr.no
        <<"\nAd \t: "<<ogr.adi
        <<"\nCins. \t: "<<ogr.cins
        <<"\nYas \t: "<<ogr.yas
        <<"\nBolum \t: "<<ogr.bolum;
    dosyaoku.close();
    system("PAUSE");
    return 0;}
```

Hata denetimi aşağıdaki şekilde de gerçekleştirilebilir.

```
ifstream dosyaoku("personel1.txt");
ios::iostate x;
x = dosyaoku.rdstate();
//bit düzeyinde and işlemi gerçekleştiriliyor
if(x & ios::failbit) cout << " Giriş/cıkış isleminde hata\n";
```

Satır Olarak Yazmak Ve Okumak

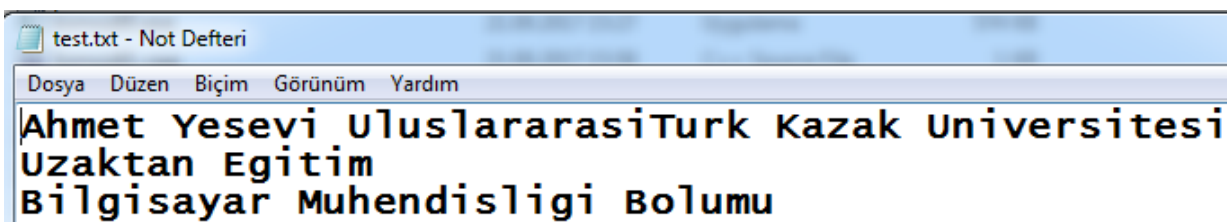
- Bir disk dosyasına veriler katarlar biçiminde kaydedilir. Bir katar içinde ve kelimeler arasında boşluklar yer alabilir. Bu verilerin kaydedilmesinde bir sorun olmaz.
- Ancak okunması sırasında sorunlarla karşılaşılacaktır. Çünkü bir değişken, sadece bir kelimenin okunmasına neden olacaktır.
- Bunun yerine, dosya içindeki bir satırın tümüyle okunarak bir değişken içine yerleştirilmesi gerekebilir.
- Böyle durumlarda getline() fonksiyonuna başvurulur. Bu fonksiyon, dosya içinde "\n" yani yeni satır karakterine ulaşıncaya dek tüm karakterleri okur.
- Söz konusu fonksiyon şu şekilde tanımlanır:

dosya.getline(tampon, tampon genişliği);

- Yukarıdaki tanıma göre getline() fonksiyonu, okuduğu satırı bir tampon alan üzerine yerleştirir. Bu alanın genişliği fonksiyon içinde belirtilir.

Örnek: Satır Kaydetme:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
string satir1="Ahmet Yesevi Uluslararası"
               "Türk Kazak Üniversitesi";
string satir2="Uzaktan Eğitim";
string satir3="Bilgisayar Mühendisliği Bölümü";
int main()
{   ofstream dosya("test.txt");
    dosya<<satir1<<endl;
    dosya<<satir2<<endl;
    dosya<<satir3<<endl;
    dosya.close();
    cout<<"Kayıt gerçekleştirildi\n\n";
    system("PAUSE");
    return 0;
}
```



Örnek: Satır Okuma:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
const int ENFAZLA=80; //yüksek olursa sorun olmaz. Düşük verirsiniz okumada sorun olur
char satir[ENFAZLA];
```

```
int main()
{ ifstream dosya("test.txt");
  while(!dosya.eof())
  { dosya.getline(satir,ENFAZLA);
    cout<<satir<<endl;
  }
  system("PAUSE");
  return 0;
}
```

```
Ahmet Yesevi Uluslararası Türk Kazak Üniversitesi
Uzaktan Eğitim
Bilgisayar Muhendisliği Bölümü
Devam etmek için bir tuşa basın . . .
```

Karakter Olarak Yazmak Ve Okumak

- Karakter katarları yanı sıra, gerektiğinde tek tek karakter giriş/çıkışları da gerçekleştirilebilir.
- Bu tür uygulamalar istream'in üyeleri olan put() ve get() fonksiyonları yardımıyla gerçekleştirilir.
- put() fonksiyonu karakterlerin kaydedilmesi, get() ise karakterlerin okunması işlemlerinde kullanılır.

Örnek:

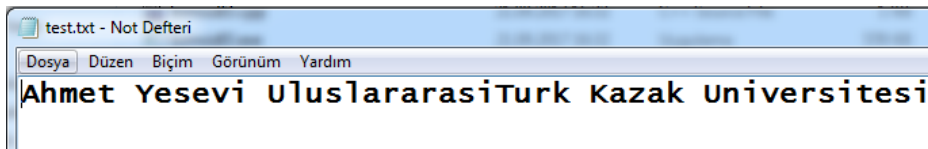
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
char karakter;
int main()
{ ifstream dosya("test.txt");
  while(!dosya.eof())
  { dosya.get(karakter);
    cout<<karakter;
  }
  system("PAUSE");
  return 0;
}
```

```
Ahmet Yesevi Uluslararası Türk Kazak Üniversitesi
Uzaktan Eğitim
Bilgisayar Muhendisliği Bölümü
Devam etmek için bir tuşa basın . . .
```


Örnek:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
string katar="Ahmet Yesevi Uluslararası"
        "Turk Kazak Universitesi";

int main()
{
    ofstream dosya("test.txt");
    for (int i=0;i<katar.size();i++)
        dosya.put(katar[i]);
    cout<<"Kayit gercekleştirildi\n\n";
    system("PAUSE");
    return 0;
}
```

**İkili Verileri Yazmak Ve Okumak**

- Önceki kısımlarda anlattığımız konular, bir disk dosyasına verilerin karakterler biçiminde kaydedilmesiyle ilgiliydi.
- Bu tür dosyalar, biçimlendirilmiş metin dosyaları olarak değerlendirilir.
- Ancak küçük boyutlu dosyalarda bu mümkün olmasına karşılık, büyük boyutlu dosyalarda bu yol tercih edilmez. Onun yerine ikili (binary) dosyalar kullanılır.
- Verileri ikili düzende kaydetmek için ofstream'ın write() fonksiyonu kullanılır. Eğer ikili verileri okumak söz konusu ise bu kez ifstream'ın read() fonksiyonu kullanılır. Bu iki fonksiyon aşağıda belirtildiği biçimde tanımlanır:

dosya.write(tampon, tampon genişliği);

dosya.read(tampon, tampon genişliği);

write()'ın görevlerini yerine getirebilmesi için ofstream'ın şu şekilde tanımlanması gerekir:

ofstream akım ("dosya adı", ios::out | ios::binary);

- Okuma işlemi gerçekleşecek ise read() fonksiyonu için aşağıdaki tanım yapılır:

ifstream akım ("dosya adı", ios::in | ios::binary);

- Tampon alana alınan verilerin karakter türünde olması gerekiyor. Eğer okunan ya da yazılan değerler karakter türü değil ise bunları karaktere çevirmek gerekir.
- Örneğin, sayi isimli değişkenin veri türü double ise, bu değişkeni tampona yerleştirmek için,

((char *)&sayi, sizeof(double))

Örnek:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
using namespace std;
char katar[]="Ahmet Yesevi Uluslararası"
           "Türk Kazak Üniversitesi";
int ogrenci_sayisi=15000;

int main()
{   ofstream dosya("test.dat",ios::out |ios::binary);
    if (!dosya)
    { cout<<"HATA:Dosya acilamadi.";
      return 1;
    }
    dosya.write(katar,strlen(katar));
    dosya.write((char *)&ogrenci_sayisi,sizeof(int));
    cout<<"Kayıt gerçekleştirildi\n\n";
    system("PAUSE");
    return 0;
}
```

Örnek:

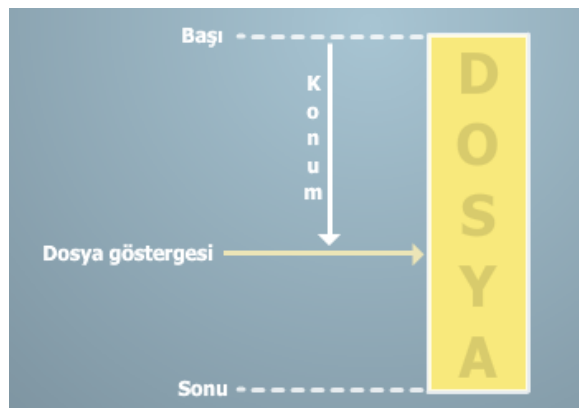
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int ogrenci_sayisi;
const int MAX=48;
char katar[MAX];

int main()
{   ifstream dosya("test.dat",ios::in |ios::binary);
    if (!dosya)
    { cout<<"HATA:Dosya acilamadi.";
      return 1;
    }

    dosya.read(katar,MAX);
    dosya.read((char *)&ogrenci_sayisi,sizeof(int));
    cout<<katar<<endl;
    cout<<"Ogrenci sayisi:"<<ogrenci_sayisi<<endl;
    system("PAUSE");
    return 0;
}
```

Rastgele Erişimli Dosyalar

- İkili verilere sahip bir dosyada verilere sırayla nasıl erişilebildiğini biliyoruz.
- Bazı durumlarda ilgili verilere doğrudan erişmek söz konusu olabilir.
- Böyle durumlarda rastgele erişim söz konusudur.
- Her dosya, okuma göstergesi ve yazma göstergesi adı verilen iki tamsayı değer ile ilişkilendirilmiştir.
- Bu değerler dosya içinde yazma ve okuma işleminin yerine getirilebileceği bayt sayısı olarak belirlenir.



- C++'da giriş ve çıkış akımlarının birer üyesi olan seekg() ve seekp() fonksiyonları kullanılarak rastgele erişim yapılır.
- **seekg() fonksiyonu**, okuma göstergesini belirlenen bir noktadan itibaren öteleme kadar hareket ettirir.
- **seekp() fonksiyonu** yazma göstergesini belirlenen noktadan itibaren öteleme kadar hareket ettirir.
- Rastgele aramalarda kullanılan seekg() fonksiyonu iki biçimde uygulanabilir.
- Bunlardan birincisi tek argüman aldığı durumdur. Eğer tek argümanı varsa, bu argüman dosyanın başından itibaren pozisyonunu gösterir.
- Eğer iki parametre varsa, birinci argüman dosyanın içinde belirli bir konumdan itibaren bir öteleme sayısını belirtir.
- İkinci argüman ise ötelemenin başlangıç konumunu belirtir.

Ötelemenin başlangıç noktası için üç seçenek vardır:

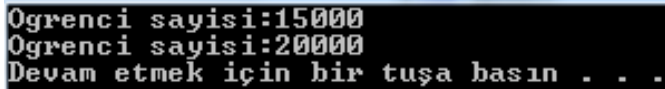
Referans	Açıklama
beg	Dosya başlangıcı
cur	Mevcut konum
end	Dosya sonu

Örneğin bir dosyanın başından itibaren 20. konumdan itibaren yazma işlemi yapılacak ise şu şekilde bir tanım yapılır:

seekp(20, ios::beg)

Örnek:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int ogrenci_sayisi=20000;
int okunan_ogrenci;
int main()
{ fstream dosya("test.dat",ios::in |ios::out|ios::binary);
  if (!dosya)
  { cout<<"HATA:Dosya acilamadi."; return 1; }
  dosya.seekg(48,ios::beg);
  dosya.read((char *)&okunan_ogrenci,sizeof(int));
  cout<<"Ogrenci sayisi:"<<okunan_ogrenci<<endl;
  dosya.seekp(48,ios::beg);
  dosya.write((char*) &ogrenci_sayisi,sizeof(int));
  dosya.seekg(48,ios::beg);
  dosya.read((char *)&okunan_ogrenci,sizeof(int));
  cout<<"Ogrenci sayisi:"<<okunan_ogrenci<<endl;
  system("PAUSE");
  return 0;}
```



Kaynakça

- Programlamaya Giriş ve Algoritmalar Ders Notları, 69/17.10.2007.
- <http://www.csharpnedir.com>
- <http://iys.inonu.edu.tr/webpanel/dosyalar/1308/file/0Algoritma.pdf>
- <http://www.xn--hakankr-fla.com.tr/Algoritma.pdf>
- <http://web.sakarya.edu.tr/~kayar/algoritma1.html>
- Yaşar,E., Algoritma ve Programlamaya Giriş, 2009
- <http://www.dahiweb.com/algoritma-ve-akis-diyagrami-ornekleri-mantiksal-akis-diyagramlari>
- <http://ceng.gazi.edu.tr/~akcayol/files/DSL5Recursion.pdf>
- <http://www.yazilimdilleri.net/YazilimMakale-1830-Kredi-Karti-Dogrulama---Luhn-Algoritmasi.aspx>
- [Algoritma ve Programlama Mantığı, H. Burak TUNGUT](#)