Programlamaya Giriş ve Algoritmalar Ders Notları

Revizyon ve Baskı Bilgisi
69/17.10.2007 01:51:28 PM

İçindekiler

Genel Programlama Bilgisi

Bir Programcının Program yazabilmesi için yapması gereken çalışmalar ve bilmesi gereken ön bilgilere ihtiyacı vardır. Çünkü bir problemin bilgisayar ortamında çözülmesi maalesef hazırlıksız olarak ve hızlı bir şekilde gerçekleşemez. Her şeyden önce programlama bir süreçtir ve programın yazılıp bitmesi ile bitmez çoğunlukla programın yaşadığı süre içerisinde değişik şekillerde devam eder.

Bu nedenle Program yazılmadan veya problem çözülmeye başlamadan önce bazı adımları sağlam atmak gerekir. Bir çok zaman bu adımlar raporlarla belgelendirilir. Çünkü yazılımın yaşam döngüsü boyunca bu raporlara veya başlangıç adımlarına ihtiyaç duyulabilir.

Bunu sağlayabilmek ve sağlam temelli programlar yazabilmek için mutlaka ön çalışmalar kağıt üzerinde gerçekleştirilir ve elde edilen donelere göre program yazılır.

Ancak programlamaya yeni iseniz hemen program yazmanız oldukça zor olacaktır. Öncelikle programlama mantığını ve problem çözme ön sezisini edinmeniz gerekecek. İşte bu ders notları siz öğrencilerin bu sezgiyi kazanmanızı ve bu mantığı oluşturmanızı sağlayacak bilgiler sunmaktadır.

Ne var ki bu ders notu sadece temel bilgileri verir. Bahsettiğimiz mantık ve sezgiler program yazdıkça gelişir. Bir çok kişi tarafından programlama yeteneğinin tanrı vergisi olduğu söylense bile (ki gerçekte doğrudur) bu yeteneği edinmek resim yapma yeteneğini edinmekten ya da güzel sanatlara karşı bir yeteneği edinmekten çok daha kolaydır.

Bu ders notları her türlü programlama dilinden bağımsız olarak hazırlanmaya çalışılmıştır. Gerekli görüldüğü yerlerde dillere ait detay veya farklılık bilgileri verilmiştir. Ancak esas olan Programlama dilinin üzerinde temel programlama mantığına sahip olmanızdır. Çünkü bu sizin Programlamacı sıfatınız için temel teşkil edecek bilgiler sunmaktadır.

Programlama dilleri zaman içerisinde gelişmiş, değişmiş, kaybolmuş veya yenileri çıkmıştır. Bu nedenle programlama bilginizi asla bir programlama diline bağlı tutmayın. Eğer Programlama mantığınız ve ön sezileriniz oldukça iyi gelişmişse, Algoritmaları kolay kurup algılayabiliyorsanız, çok karmaşık sorunlar üzerinde fikir yürütüp çözüm üretebiliyorsanız bildiğiniz programlama dilinin fazlaca bir önemi kalmamaktadır. çünkü çoğunlukla 1-2 hafta gibi bir sürede bir programlama dilini orta düzeyde öğrenebilirsiniz. En doğrusu da tabii ki her derde çözüm olabilecek temel bir programlama dilini çok iyi bir şekilde öğrenip her çözüme uyarlayabilmeniz. Ancak önemli olan Programlama mantığını iyi kapmış olmanız.

Dikkat ettiyseniz Programlama dilinde alternatiflerden bahsettik ancak Programlamanın alternatifi bulunmamaktadır. Yani bir program yazılacaksa mutlaka programlamanın kurallarına göre yazılmalıdır.

Tanımlar

Bu ders notlarının daha iyi anlaşılabilmesi için şu tanım ve kavramların bilinmesinde fayda bulunmaktadır.

Bilgisayar Nedir?

Verileri İşleyerek Özet bilgiler şekline sokabilen, bu veri ve bilgileri yüksek kapasitelerde saklayıp başka ortamlara iletebilen elektronik cihazlardır.

Bu işlemleri yaparken çok yüksek hızlarda ve bıkmadan-usanmadan tekrarlı olarak aynı işleri yapan bir cihazdır.

Ancak tek başına bir Bilgisayar donanımı bu işleri gerçekleştirecek durumda değildir. Bu işlemler donanım tarafından yapılsa da asıl iş yazılımlar tarafından gerçekleştirilmektedir.

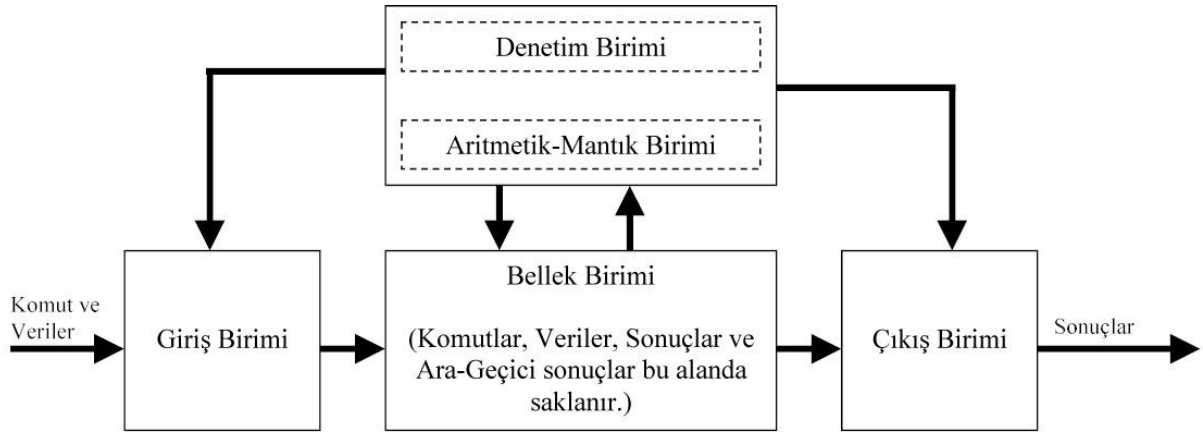
Ancak Yukarıdaki Çizimde görülen her bir adımın gerçekleştiren yazılımdır.

Bilgisayar Organizasyonu?

Bir Elektronik Bilgisayar yukarı şekildeki gibi temel birimlere sahiptir. Bu şekilde bir önceki konuda bahsettiğimiz giriş, çıkış işlemleri, veri işleme ve veri saklama işlemleri gerçekleştirilir.

Giriş Birimi	Klavye, Fare, mikrofon, kamera vb. cihazlardan biri veya daha fazlasından oluşur. Veri ve komutların bilgisayara yollanmasını sağlar.
Bellek Birimi	Veri, komut ve programların saklandığı donanımları temsil eder. Ana Bellek çoğunlukla RAM olarak anılır ve SIMM gibi donanımlarla belirlenir. Yan veya İkincil bellek ise Floppy ve sabit diskleri ifade eder ve bilgilerin daha kalıcı saklanmasını sağlayan alanları temsil eder.

Aritmetik-Mantık Birimi	C P U	Bilgisayardaki tüm Aritmetik ve Mantıksal işlemlerin yapıldığı birimdir. Tüm işlemleri yapan ana birimdir.
Denetim Birimi		Bilgisayar programının çalışmasını denetleyen, programda yapılan hesaplamalara göre hangi işlerin yapılacağına karar veren birimdir. Makine diline çevrilmiş bir programda komutları teker teker ve sırayla Ana işlem birimine getirip yorumlar ve sonuçta komutu çalıştırır.
Çıkış Birimi		Bilgisayar ortamında oluşan verilerin dış ortama verilmesini sağlayan birimlerdir. Ekran, yazıcı gibi birimlerdir.



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.

Bilgisayarın Tarihçesi

Bu konuda çok değişik ve çeşitli kaynaklar bulmak mümkün olabilir ancak şöyle basit bir kronolojik sıra geçmek mümkündür

M.Ö. 500	Abaküs
1642	Pascalın Mekanik Toplama Makinesi
1827	Babbage'in çıkarma makinesi
1941	İkili>İkili Mekanik Hesaplayıcı (Zuse)
1944	Ondalık Elektromekanik Hesaplayıcı (Aiken)
1945-54	İlk Kuşak Vakum Tüpler ve ışınlar
1955-64	İkinci Kuşak Tranzistörler ve Manyetik Bellekler
1965-71	Üçüncü Kuşak Tümleşik Devreler
1971-90	Dördüncü Kuşak VLSI Devreler

1982	IBM Pc & MS-DOS
1984	MAC
1990'lar	Paralel İşlemciler, Yapay Zeka, İnternet ve WWW

Bilgisayar Türleri

Bilgisayarlar kullanım amaçlarına göre büyüklük ve kapasite bakımında şu şekilde sınıflandırılabilirler.

- Süper Bilgisayarlar
- Mainframe tipi Bilgisayarlar
- Workstation tipi Bilgisayarlar (İş İstasyonları)
- Mikro Bilgisayarlar
- Kişisel Bilgisayarlar

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. Gerekiyorsa 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.

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 gerçekte ilk paragrafta anlatılan kısım çoğunlukla sistem analizi veya sistem çözümleme olarak anlatılır. Diğer adımlar Programlama diye tanımlanabilir. 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 Çizgesi 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.

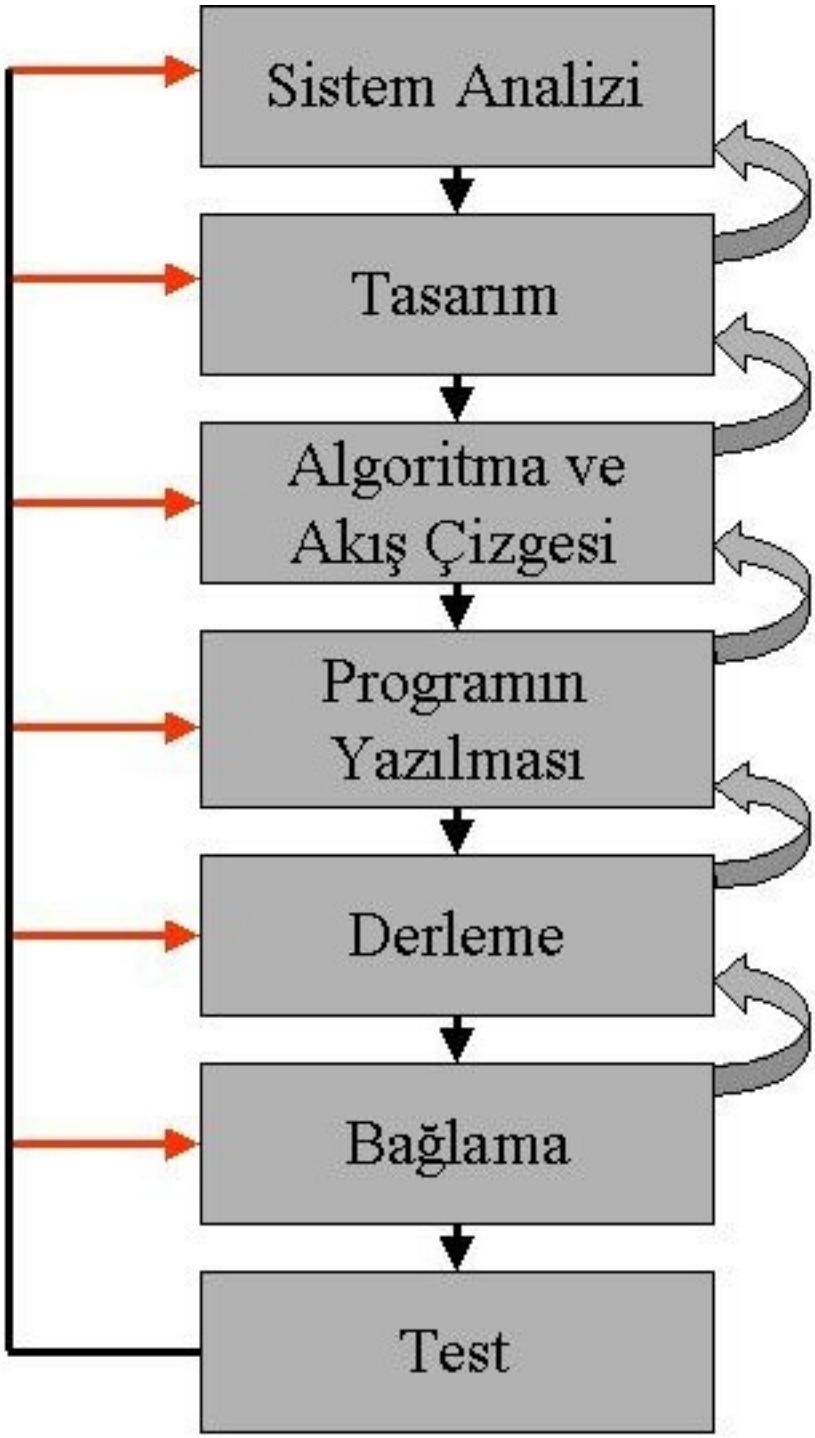
Her Programlama dili için bir derleyici olması gerekmektedir.

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.

Yazılım Geliştirme

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



Bu çizgeden anlaşılacağı gibi adımlardan birinde bir sorunla karşılaşılırsa 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 değişkenler 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.

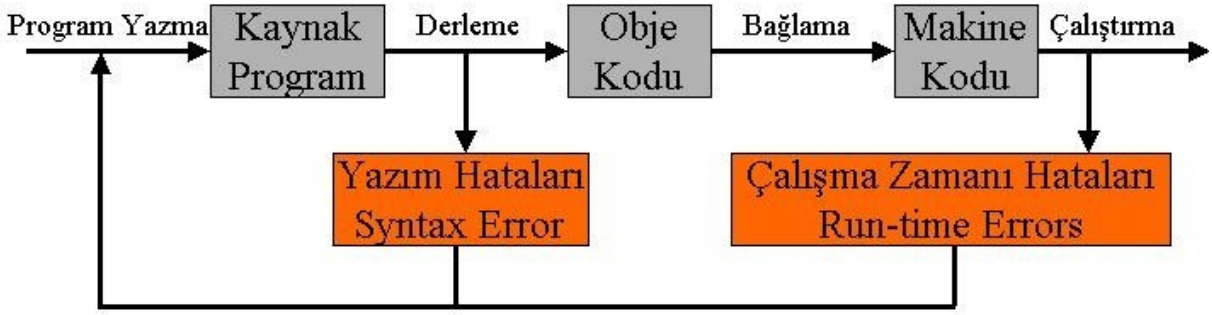
Derleme : Programlama Dili ile yazılmış programın yazım hatalarının olup olmadığının kontrol edilmesini ve ara kod olarak Obje kodun üretilmesini sağlama adımıdır.

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

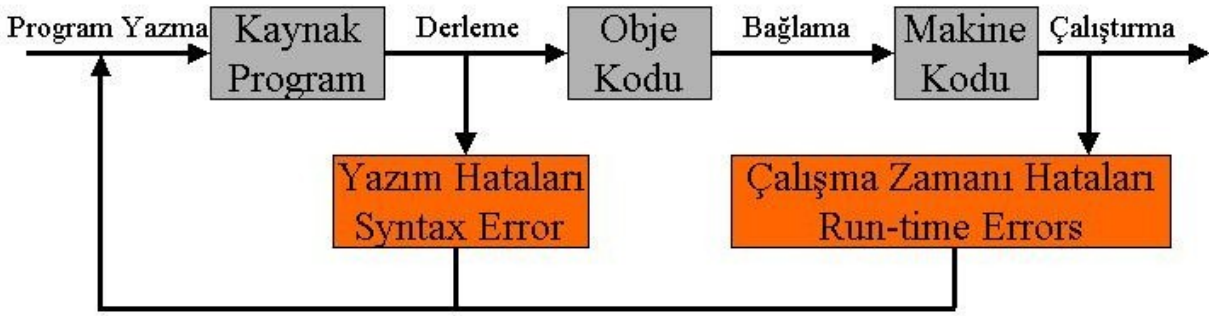
Çalıştırma : Oluşturulan Makine dili Programının çalıştırılması adımıdır. Yukarıdaki adımların hepsi yolunda gittiyse program sorunsuz olarak çalışabilmelidir.

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

Yaşam Döngüsünün Sağlanması : Yukarıdaki Akış Çizgesi dikkat edilirse aslında bir döngüdür. Hatta test aşamasında sorun çıkmazsa bile Sorunun tanımında yani ihtiyaçlarda bazı değişiklikler olursa adımlar baştan aşağı tekrar incelenmek zorunda kalınır. Bu çizgeye bir Yazılımımın Yaşam Döngüsü de denilebilir. Bu çizimde Yazılımın Bakım süreci göz önüne alınmamıştır.



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. Düzeltilmesi 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 üretilememiştir 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 : Karşılaşabileceğiniz 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 komut veya yaz

Algoritmalar

Bir bilgisayar programı aslında sıra düzensel olarak tanımlanmış bir dizi komuttan başka bir şey değildir. Bu açıdan bizim yazmaya çalışacağımız programda bir dizi komut yani eylem topluluğudur. Her programda bu eylemler yazıldıkları sırada gerçekleştirilir veya çalıştırılırlar. Aslında bizim günlük hayattaki yaşantı tarzımız dahi düzenli olarak bir takım işlemlerin sıra ile yapılması şeklindedir. Yani bir iş yapabilmek için bir takım alt iş veya olayları peş peşe gerçekleştiririz.

Algoritmanın tanımını daha önce vermiştik burada bu tanımı tekrar etmek faydalı olacaktır. ***Bir sorunu çözebilmek için gerekli olan sıralı mantıksal adımların tümüne algoritma*** *denir.* Bir algoritmadan beklenen bir takım özellikler olduğunu da yine daha önceki tanımlar bölümünde bahsetmiştik. Biz şimdi mümkün olduğu kadar bu tanım ve özelliklerden yola çıkarak örneklerle bir kaç algoritma vermeye çalışacağız.

Öncelikle bir ev hanımının pasta yapmak istediğini varsayalım. Bu pastanın yapılabilmesi için gerekli bir takım işlemler ve alt adımlar bellidir. bir ev hanımı da sıra ile bu adımları uygulayarak bu pastayı yapar. Şöyle ki:

1. Pastanın yapımı için gerekli malzemeleri hazırla
2. Yağı bir kaba koy
3. Şekeri aynı kaba yağın üzerine koy
4. Yağ ve şekeri çırp
5. Karışımın üzerine yumurtayı kır
6. Tekrar çırp
7. Kıvama geldi mi diye kontrol et
8. a. Kıvamlı ise 9. adıma devam et
   b. Değilse 6. adıma dön.
9. Karışıma un koy
10. Karışıma vanilya, kabartma tozu vb. koy
11. Karışımı Kıvama gelinceye kadar çırp
12. Pastayı Kek kalıbına koy
13. Yeteri kadar ısınan fırına pastayı koy
14. Pişimi diye kontrol et
15. a. Pişmiş ise 16. adıma devam et
    b. Değilse 14. adıma dön
16. Keki fırından çıkart
17. Fırını kapat
18. Kekin ı kapat
19. Kekin soğumasını bekle
20. Keki servis edebilirsin.

Bu algoritma günlük hayattan bir örnek. Gerçekte biz her işimizi algoritmik olarak yaparız ancak bunu farkına varmayız. Yukarıdaki algoritmayı inceleyecek olursak bir kekin yapılması için gerekli tüm adımlar sıra ile yer almış durumda. Gerçi algoritma anlatacağımız konuların daha iyi anlaşılabilmesi için biraz farklı ele alınmıştır ama gerçek bir Pasta yapım aşamasını içerir. Bu algoritma ve diğer tüm algoritmalar için bilmemiz gereken bazı konular bulunmaktadır:

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.

Algoritmada algoritmanın genel işleyişini etkileyebilecek hiç bir belirsizlik olmamalıdır. (Bu örnekte öyle bir belirsizlik var. Bir fırının yeteri kadar ısına bilmesi hangi koşula bağlıdır, bu fırın ne zaman açılmış olmalıdır ve kaç dereceye ayarlanmış olmalıdır. gibi...)

Algoritmada bazı adımlar yer değiştirebilir . Ancak bir çok adımın kesinlikle yer değiştiremeyeceğini bilmeliyiz. Yanlış sıradaki adımlar algoritmanın yanlış çalışmasına neden olacaktır. (9 ve 10. adım değiştirilebilir. 2-3. adımlar yer adımlar yer değiştirebilir.) Ancak 13-16. adımlar kesinlikle yer değiştiremezler.

Peki Bilgisayarda çözülecek bir sorunu nasıl algoritma ile ifade ederiz? Bunun için öncelikle bir sorun tanımlayalım. Başlangıç ta basit olması için şöyle bir problem üzerinde düşünelim: *Bilgisayara verilecek iki sayıyı toplayıp sonucu ekrana yazacak bir program için algoritma geliştirmek isteyelim.* Sorun son derece basit ancak sistem tasarımının net yapılabilmesi için sorun hakkında anlaşılamayan tüm belirsiz noktalar açıklığa kavuşturulmalıdır. Örneğin sayılar bilgisayara nereden verilecek, Klavye, Dosya veya belki başka bir ortam. Bu ve buna benzer soru ve tereddütleriniz varsa sorun sahibine bunları sormalı ve sistem analizi yapmalısınız.

Sonra bulacağımız çözümü algoritma haline dönüştürebiliriz.

1. BAŞLA
2. A sayısını oku
3. B sayısını oku
4. TOPLAM=A + B işlemini yap
5. TOPLAM değerini ekrana yaz

6. SON

Biraz daha karmaşık bir sorun şöyle olsun: *Klavyeden girilecek iki sayıdan büyük olanından küçük olanını çıkarıp sonucu ekrana yazacak program için bir algoritma geliştiriniz.*

A. BAŞLA
B. A sayısını oku
C. B sayısını oku
D. Eğer A büyüktür B SONUC=A-B
   Değilse SONUC=B-A
E. SONUC değerini ekrana yaz

F. SON

Gerçekte bir algoritma genellikle üç ana bölümden oluşur. Bunlar :

- Algoritmanın giriş ve ilk işlemlerinin yapıldığı bölüm
- Döngüsel bir bölümün olduğu kesim. Bu bölüm iterasyonlarla bir işlemin sürekli tekrar edilerek sonuca ulaşmayı sağlayan bölümdür.
- Son işlemlerin yapıldığı bölümdür. Bu bölüm elde edilen sonuçların ekrana yazılmasını sağlayan bölümdür.

Örneğin birden Klavyeden girilen bir n değerine kadar sayıları toplayan ve sonucu ekrana yazan bir algoritmayı geliştirelim.

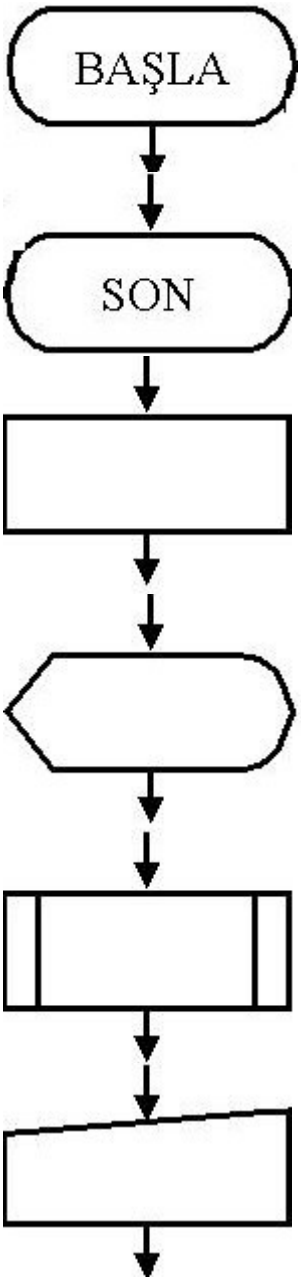
1. BAŞLA
2. N OKU
3. T=0
4. X=1
5. T=T+X
6. X=X+1
7. EĞER X<=N İSE 5. ADIMA GİT

8. T YAZ

Bu algoritmalar oldukça basit algoritmalar olup algoritma kavramının yerleşmesini sağlayan örneklerdir.

Akış Çizgeleri

Bir algoritmanın şekillerle görsel gösterimidir. Dikkat edildiyse algoritma doğal dille yazıldığı için herkes tarafından anlaşılamayabilir ya da istenmese de başka anlamlar çıkarılabilir. Ancak akış çizgelerinde her bir şekil standart bir anlam taşıdığı için farklı yorumlanıp anlaşılamaması mümkün değildir. Bir algoritmanın ifade edilebilmesi için kullanılan şekiller ve anlamları şunlardır:



Bir algoritmanın başladığı konumu göstermektedir. Tek çıkışlı bir şekildir.

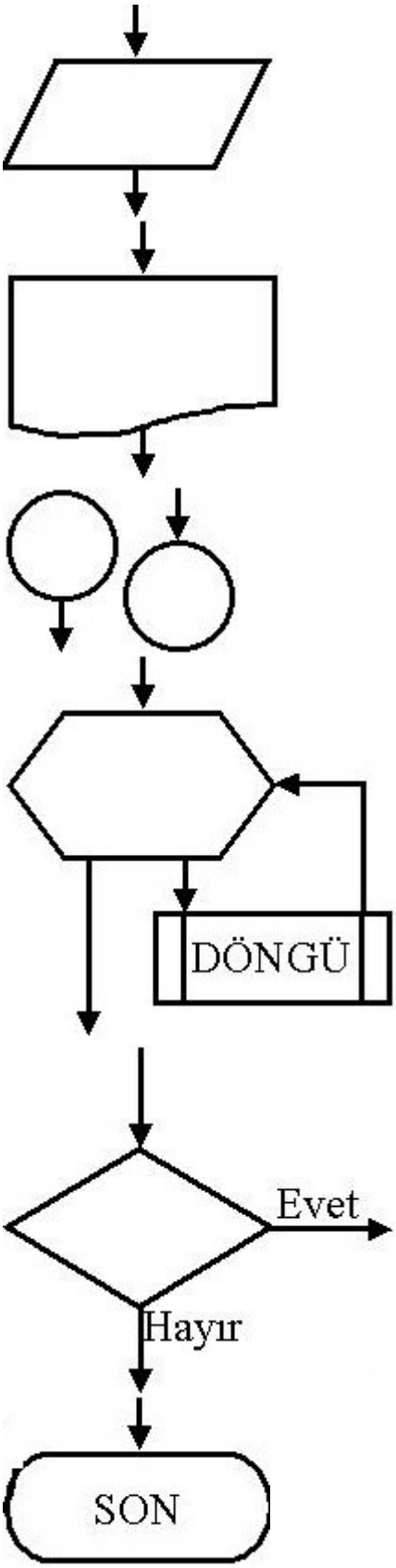
Bir algoritmanın bittiği konumu göstermektedir. Tek girişli bir şekildir.

Bir algoritmada aritmetik işlem yapılmasını sağlayan şekildir. Bu dörtgen kutu içerisine yapılmak istenen işlem yazılır. Tek girişli ve tek çıkışlı bir şekildir.

Algoritmada bir bilginin ekrana yazılacağı konumu gösteren şekildir. Ekrana yazılacak ifade ya da değişken bu şekil içerisine yazılır.

Bir algoritmada başka bir yerde tanımlanmış bloğun yerleştiği konumu gösteren şekildir. Kutu içerisine bloğun adı yazılabilir.

Klavyeden Bilgisayara bilgi girilecek konumu belirten şekildir. Girilecek bilginin hangi değişkene okunacağını kutu içerisine yazabilirsiniz.



Giriş Çıkış komutunun kullanılacağı yeri belirler ve kutu içerisine hangi değişkeni ve OKU mamı YAZ mı yapılacağını belirtmeniz gerekir

Bilginin Yazıcıya yazılacağı konumu gösteren şekildir.

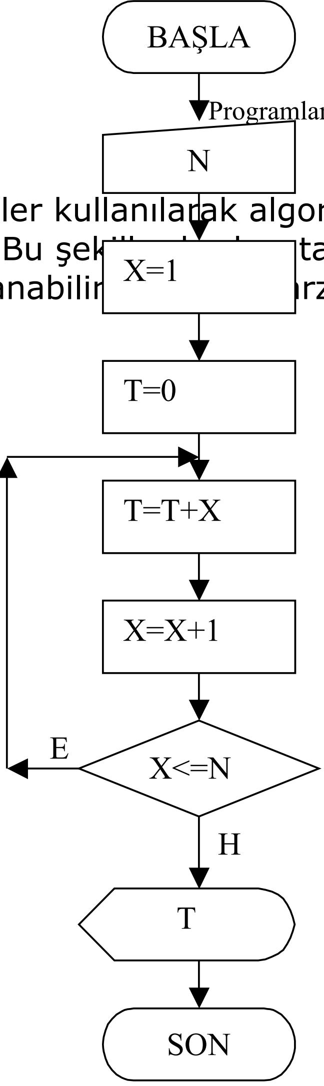
Bir algoritmanın birden fazla alana yayılması durumunda bağlantı noktalarını gösteren şekildir. Tek girişli veya tek çıkışlı olarak kullanılırlar.

Bir işlemin belli bir sayıda veya belli bir koşul doğru olduğu sürece tekrar edilmesini sağlayan döngü komutunu gösteren şekildir. Bu döngüde altıgen içerisine ya koşul yada döngünün başlangıç, adım ve sonlanma değerlerini belirtebilirsiniz. DÖNGÜ olarak belirlenen blokta da tekrar edilmek istenen komutlar yer almaktadır.

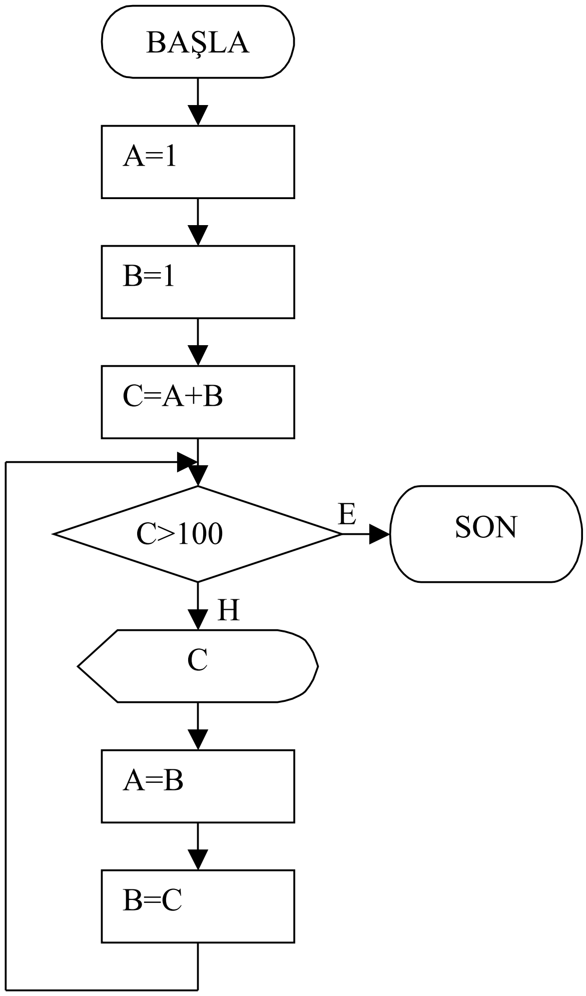
Bir algoritmada bir kararın verilmesini ve bu karara göre iki seçenekten birinin uygulanmasını sağlayan şekildir. burada eşkenar dörtgen içerisine kontrol edilecek mantıksal koşul yazılır. Program akışı sırasında koşulun doğru olması durumunda "Evet" yazılan kısma Yanlış olması durumunda "Hayır" yazılan kısma sapılır. Tek girişli ve çift çıkışlı bir şekildir.

Programın bittiği yer ya da yerleri gösteren bir şekildir.

Bu şekiller kullanılarak algoritma ile oluşturulan çözümler akış çizgelerine çevrilir. Bu şek[illegible] tarafından anlaşılabilir ve doğru olarak yorumlanabilir[illegible]rz ederler.



Bu örnekte 1'den N değerine kadar sayıları toplayıp ekrana yazan bir akış çizgesi çizilmiştir.



100'e kadar sayılardan fibonecci dizisinin elemanlarını listeleyen akış çizgesidir.

Aritmetik ve Mantıksal ifadeler

Bir takım işlemlerin yapılabilmesi için aritmetik işlemlerin nasıl tanımlandığını bilmeniz gerekecek. Program akışının doğru bir şekilde sağlanabilmesi için bazı yerlerde karar verilmesi gereklidir. Bu kararların verilmesini sağlayacak Mantıksal ifadeler bilgisayarlar tarafından yapılabilmelidir.

Aritmetik İfadeler

Bir programın istenilen işleri yerine getirebilmesi için hesap yapabilmesi ve bu hesaplar sonucunda elde edilen değerleri saklayabilmesi gerekmektedir. Genel olarak tüm programlama dillerinde aritmetik işlemler şu şekilde tanımlanmıştır:

```
Değer|Değişken {aritmetik operatör} Değer|Değişken
```

Bu tip bir gösterimde Değer denilen kesim sabit bir değeri temsil eder. Değişken ise içerisinde her an farklı bir değer tutabilecek bir tanım ifade eder. Aritmetik Operatör ise aritmetik işlemin özünü oluşturan bir işleçtir.

Aritmetik Operatör

Kısaca dört işlem olarak da ifade edebileceğimiz işleçlerdir. Çoğunlukla dört tanedirler ve cebirde kullanılan öncelikleri aynı şekilde kullanırlar.

*	Çarpma işlemini gösteren işleçtir. Bölme ile eş önceliklidir.
/	Bölme işlemini gösteren işleçtir. Çarpma ile eş önceliklidir.
+	Toplama işlemini gösteren işleçtir. Çıkarma ile eş önceliklidir.
-	Çıkarma işlemini gösteren işleçtir. Toplama ile eş önceliklidir.

Temel olarak bu işleçler her programlama dilinde bulunur. öncelikleri yukarıdaki sırada olduğu gibidir. Eş öncelikli aritmetik ifadeler bulunursa bu ifadelerde işlem soldan sağa doğru yapılır.

Bu işlem önceliklerini değiştirmek için her programlama dilinde "(", ")" (Parantez aç-kapa) ifadelerini bulabilirsiniz.

Bunlardan hariç olarak bazen üz alma operatörlerinden bahsedilebilir. bazen de mod operatörü ve tam bölme operatörü gibi değişik ve standart olmayan operatörler yar alabiliyor.

Aritmetik Operand

Aritmetik ifadelerde işleme giren tarafların her birine aritmetik operand yani aritmetik işlenen denir. yukarıdaki tanımda Değer|Değişken ikilisinin her biri işlenen yani operand tanımını sağlar. Bir aritmetik ifade de bir çift operand ve bir operatör minimum olması gereken ifadelerdir. Ancak daha fazla aritmetik işlenen ve işleç olabilecektir. Örneğin A * B + (12 + B) işlemi de bir aritmetik ifadedir.

Değer Aktarma deyimi

Genellikle bir aritmetik ifade ile hesaplanan değer başka bir değişkene aktarılır. Bu işlem çoğunlukla "=" sembolü ile gerçekleştirilir. Ancak başka bir komut, sembol veya ifade ile de bu değer aktarılması yapılabilir.

Mantıksal ifadeler

Bir değerin başka bir değer ile karşılaştırılması sonucu doğru veya yanlış sonuç elde edebilen ifadelerdir.

Mantıksal operatörler ve karşılaştırma operatörlerinin değişik alternatiflerle bir araya gelmesinden oluşur ve şu şekilde ifade edilir:

```
Değer|Değişken|Aritmetik ifade {Karşılaştırma Operatörü}
                Değer|Değişken|Aritmetik ifade
Karşılaştırma Değimi {Mantıksal operatör} Karşılaştırma Değimi
```

Karşılaştırma Operatörü

İki değer veya aritmetik ifadeyi bir biriyle karşılaştırmayı sağlayan sembollerdir.

=	İki değerin eşit olup olmadığı karşılaştıran işleçtir.
>, >=	İki değerden soldakinin sağdakine oranla büyük olup olmadığını veya büyük eşit olup olmadığı kontrol eden operatördür.
<, <=	İki değerden soldakinin sağdakine oranla küçük olup olmadığını veya küçük eşit olup olmadığı kontrol eden operatördür.
<>	İki değerin farklı olup olmadığını karşılaştırılan işleçtir.

A>=8 ifadesi a değişkeni içindeki değeri 8 den büyük mü diye karşılaştırır.
A<>12 ifadesi a değişkeni içindeki değer 12'den farklı mı diye karşılaştırır.

Mantıksal Operatör

Birden fazla mantıksal karşılaştırma deyiminin birleştirilmesini sağlayan işleçlerdir. VE, VEYA gibi ifadelerdir. Bu operatörler mantıksal ifadelerden iki veya daha fazlasının mantıksal doğruluk tablolarına göre birleştirilmesini sağlar.

VEYA için doğruluk tablosu

P	**Q**	**P VEYA Q**
D	D	D
D	Y	D
Y	D	D
Y	Y	Y

VE için doğruluk tablosu

P	Q	P VE Q
D	D	D
D	Y	Y
Y	D	Y
Y	Y	Y

Son mantıksal operatör DEĞİL operatörü olup tek işlenen alır ve aldığı işlenenin mantıksal değilini alır. Yani doğru olan değeri yanlışa yanlış olan değeri doğruya çevirir.

P	**DEĞİL(P)**
D	Y
Y	D

Örneğin bir değişken içindeki değerin 1 ile 10 arasında olup olmadığını kontrol eden ifade şu şekildedir: (A>=1 VE A<=10). Bu tür ifadeler yazılırken dikkat edilmelidir örneğin (A>=1 VE A<=10) yerine yanlışlıkla (A>=1 VEYA A<=10) yazılması her zaman doğru sonuç üretecektir. Yada (A=3 VE A=8) gibi bir ifade de asla doğru olamayacaktır.

Değişken ve Dizi Tanımlama

Bilgisayarda yapılacak hesaplar sonucu elde edilecek değerleri ve dışarıdan bilgisayara girilecek değerler değişkenlerde tutulur. Değişkenler içlerinde tuttukları değerlerin türlerine göre sınıflandırılır. Bu değişkenler bazen basit şekilde olmaz kayıt yapısında ya da dizi yapısında olabilirler.

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.

Tam Sayı Sabitleri

Sadece rakamlardan oluşan ifadelerdir. Bu tip sayıların içerisinde sadece rakam ve gerekiyorsa işaret sembolü olabilir. Aritmetik işlemlere girebilirler. (12, 1233, -3422 gibi)

Kesirli Sayı Sabitleri

Rakam ve ondalık ayıracı olarak . (nokta) sembolü kullanılabilir. ayrıca işaret sembolü olarak (-) kullanılabilir. Aritmetik işlemlere girebilirler. (3.14, 2.41. -2.11221 gibi)

Dizgi Sabitleri

Tırnak içerisinde yazılan her ifade dizgi sabiti olarak algılanır. Programlama dillerinde farklı tırnak sembolleri kullanılabilir. Çoğunlukla " sembolü tercih edilir. Aritmetik işlemlere giremez. ('Malatya', "Merhaba", "Dünya", "12322" gibi).

Değişken Nedir?

Her seferinde farklı değerler içerebilen yapıdır. Programlarda değişkenleri kutu olarak sembolize ederek çözümlemek mümkündür. Değişkenler de sabitlerde olduğu gibi türlere sahiptir. Sabitlerdeki türler aynen geçerlidir.

Dizi Nedir?

Bazı durumlarda kullanım amacı aynı olan birden fazla hatta oldukça fazla sayıda değişkene ihtiyaç bulunur. Bu tip bir durumda bu değişkenler tek tek tanımlanmak yerine bir ad altında indisle tanımlanırlar. bu tanıma dizi denir. Bazı sorunların çözümü için bu tanımlar zorunludur. Mesela 1000

sayıyı sıralamak için basit değişken tanımlama işlemi başarısız olacaktır. Veya bir sınıf listesinin tümünü bilgisayarda tutmak isterseniz basit değişkenler uygun değildir.

Çoğunlukla bu tip bir tanımda dizinin bir adı vardır. Erişmek istediğiniz dizinin eleman indis değeri ile bu dizi içerisinde istediğiniz elemana ulaşabilirsiniz.

A dizisi

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)

Bir dizi bu şekilde düşünülebilir. Her bir eleman yukarıdaki gibi isimlendirilir. Bunun bize sağlayacağı avantaj indis değerlerinin değişkenlerle ifade edilebilmesinde yatmaktadır. Yani tek bir satır komut ile tüm dizi elemanları mesela ekranda yazdırılabilir.

Algoritma Çalıştırma

Bazen Bir takım algoritmaların ne işe yaradığını anlamak veya algoritmanın doğru çalışıp çalışmadığını test etmek için algoritmayı çalıştırmak gereklidir. Algoritmayı çalıştırmak demek algoritmanın adımlarını sıra ile uygulamak, oluşan değişken değerlerini bir tablo üzerinde göstermek demektir.

1. BAŞLA
2. A OKU
3. B OKU
4. C OKU
5. TOP=0
6. SAY=A
7. TOP = TOP+SAY
8. SAY=SAY+C
9. EĞER SAY<=B İSE 7. ADIMA GİT
10. TOP YAZ

11. SON

Şeklinde verilmiş bir algoritmamız olsun. Bu algoritma için A→3, B→12 ve C→2 değerleri girilince SAY ve TOP değişkenlerinde hangi değerlerin oluşacağını algoritmayı adımlayarak gösterelim.

Değişkenlerin Her birinin Değeri					**Açıklama**
A	**B**	**C**	**TOP**	**SAY**	
3	12	2	0	3	6. adıma kadar programın ilk çalıştırılışında değişkenlerin elde ettiği değer
			3	5	7. ve 8. adımların çalıştırılmasından sonraki değerler
			8	7	7. ve 8. değerler tekrar çalıştırılıyor
			15	9	9<=12 olduğu için 7. ve 8. tekrar çalıştırılıyor.
			24	11	11<=12 olduğu için 7. ve 8. tekrar çalıştırılıyor.
			35	13	13<=12 olmadığı için algoritma 10. satırdan çalışmaya devam edecektir. Ve 10. satırdaki ifadeden dolayı ekrana 35 değeri yazılacaktır.

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 sıfırıncı basamak olmak kaydıyla tüm basamaklarımız sola doğru numaralandırılır. Sonra her basamaktaki sayısal değeri $2^{basamak}$ değeri ile çarpar ve bulunan tüm değerleri toplarız.

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

$=\mathbf{1*2^0} + 1*2^1 + \mathbf{0*2^2} + 1*2^3 + \mathbf{1*2^4} + 0*2^5 + \mathbf{0*2^6} + 1*2^7$

$=\mathbf{1} + 2 + \mathbf{0} + 8 + \mathbf{16} + 0 + \mathbf{0} + 128$

$=155$

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 sayı sistemidir. Bunun gibi bir programcının bilmesi gerekebilecek 16'li 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.

Veri İşleme

Bilgisayarımızda anlatmak, görmek, duymak istediğimiz her bilgi veya medya bilgisayar ortamında sayısal olarak saklanır. Bunların üzerinde yapacağınız işlemlerin çoğu da aslında matematiksel veya mantıksal işlemlerdir. Örneğin bir çizim bilgisayarda nokta nokta saklanır. İki boyutlu bir düzlem üzerinde (0,0) koordinatındaki noktanın rengi sayısal olarak verilir. Örneğin bu değer 0 ise renk siyah 65555 ise renk beyazdır. Bu sayısal değeri değiştirerek (0,0) koordinatındaki noktanın rengini değiştirmiş olursunuz.

Aynen bu şekilde tüm bilgiler bilgisayar ortamında uygun sayısal veriler şeklinde tutulur. Yapılacak işlemler bu verilerin üzerinde matematiksel ya da mantıksal yollarla yapılan işlemlerdir. Bir resmin büyütülüp küçültülmesi resimlerin üzerine efekt eklenmesi, seslere efektlerin eklenmesi, video görüntülerinde istenilen ekleme ve çıkarmaların yapılması, bir nesnenin üç boyutlu görüntüsünün elde edilip hareket kazandırılması gibi onlarca konu veri işleme esaslarına dayanan hesaplamalarla elde edilir. Bu tip kavramların her biri için internet'ten algoritma vb. aratılıp kullanılabilir.

Algoritma Örnekleri ve Sorular

1. **Soru**

Klavyeden girilecek X değerinden N değerine kadar tüm doğal sayıları listeleyen algoritmayı geliştiriniz.

Çözüm: X değişkeni ve N değişkeni klavyeden girilecek olup X'den N'e kadar elde edilecek her bir değer ekrana yazılacaktır.

```
1. X OKU
2. N OKU
3. X YAZ
4. X=X+1
5. EĞER X<=N İSE 3. ADIMA GİT

6. SON
```

2. **Soru**

Klavyeden girilecek bir N değerine kadar fibonecci dizisini bulan algoritmayı geliştiriniz. Fibonecci dizisi 1 1 değerleri ile başlar ve yeni değer kendinden öndeki iki değerin toplamı olarak bulunur. (1 1 2 3 5 8 13 21...)

Çözüm: Fibonecci dizisi a,b,c şeklinde 3 değişken ile hesaplanabilecek bir dizidir. Çünkü sürekli a ile b toplanarak c üretilir ve sonra bu a, b, c değişkenleri kaydırılır.

```
7. N OKU
8. A=1
9. B=1
10.A,B YAZ
11.C=A+B
12.EĞER C>N İSE 11. GİT
13.C YAZ
14.A=B
15.B=C
16.5. ADIMA GİT

17.SON
```

3. **Soru**

Klavyeden girilen bir sayının tüm tam bölenlerini bulup listeleyen (Ekrana yazan) bir algoritma geliştiriniz.

Çözüm: Bu örneği çözebilmek için böleni kalma operatörünün varlığını kabul edeceğiz. Bu amaçla Cdilinde de kullanılan % operatörünü kalanı bulma operatörü olarak kullanacağız.

```
18.N OKU
19.X=1
20.EĞER N%X=0 İSE X YAZ
21.X=X+1
22.EĞER X<=N İSE 3. ADIMA GİT

23.SON
```

4. Soru

Klavyeden girilen üç sayıdan büyüklük sıralamasına göre ortadakini bulup ekrana yazan program için algoritma yazınız.

Çözüm: Bu algoritma üç sayının klasik yollarla sıralanmasını ya da büyüklük sıralamasının bulunmasının ne denli zor olduğunu anlatmaya çalışan bir örektir.

```
24.A, B, C OKU
25.EĞER A>B VE B>C İSE B YAZ
26.EĞER C>B VE B>A İSE B YAZ
27.EĞER B>A VE A>C İSE A YAZ
28.EĞER C>A VE A>B İSE A YAZ
29.EĞER A>C VE C>B İSE C YAZ
30.EĞER B>C VE C>A İSE C YAZ

31.SON
```

5. Soru

Klavyeden girilen A ve B gibi iki sayının bölme işlemi kullanmadan sadece toplama ve çıkarma kullanarak kalanlı bölme yapan algoritmayı yazınız.

Çözüm: Bu örnek çok eski işlemcilerde çarpma işleminin tanımlı olmadığı durumlar için çarpma ya da bölme yapmak amacıyla kullanılan algoritma olarak karşımıza çıkmıştır. İlk okulda da fasulye hesabına dayanarak çarpmayı öğrendiğimiz yılları hatırlamamızı sağlayabilir.

```
32.A,B OKU
33.BOLUM=0
34.KALAN=0
35.EĞER A<B İSE KALAN=A, 8. ADIMA GİT
36.A=A-B
37.BOLUM=BOLUM+1
38.4. ADIMA GİT
```

```
39.BOLUM, KALAN YAZ
```

6. Soru

Klavyeden girilen A ve B gibi iki sayıyı, çarpma işlemi kullanmadan sadece toplama ve çıkarma kullanarak çarpıp sonucu ekrana yazan algoritmayı yazınız.

Çözüm:

```
40.A, B OKU
41.SAY=0
42.TOPLA=0
43.EĞER SAY>=B İSE 8. ADIMA GİT
44.TOPLA=TOPLA+A
45.SAY=SAY+1
46.4. ADIMA GİT

47.TOPLA YAZ
```

7. Soru

Sıfır –0 girilinceye kadar klavyeden okutulan değerlerin ortalamasını hesaplayıp ekrana yazan algoritmayı geliştiriniz

Çözüm: Bu örnekte bir toplam değerini tutabilecek bir de sayıları sayabilecek iki değişkene ihtiyaç vardır. Klavyeden girilen her değer sıfır ile karşılaştırılacak değilse işleme devam edilecektir. Sıfır ise ortalama hesaplanıp ekrana yazılacaktır.

```
48.T=0
49.SAY=0
50.X OKU
51.EĞER X=0 İSE 8. ADIMA GİT
52.T=T+X
53.SAY=SAY+X
54.3. ADIMA GİT
55.ORT=T / SAY

56.ORT YAZ
```

8. Soru

Klavyeden girilecek 20 sayının tek olanlarını ayrı çift olanlarını ayrı toplayıp sonuçları ekrana yazan algoritmayı geliştiriniz

Çözüm: Bu algoritmada yine % operatörünün kalanı bulan operatör olarak kullanılacağını varsayıyoruz. Çünkü bu algoritmada bir sayının çift mi tek

mi olduğunu anlamak ancak ve ancak o sayının ikiye bölümünden kalanın 1 mi 0 mı olduğuna bağlıdır

```
57.CIFT=0
58.TEK=0
59.SAY=0
60.X OKU
61.SAY=SAY+1
62.EĞER X%2=0 İSE CIFT=CIFT+X
       DEĞİLSE TEK=TEK+X
63.SAY=SAY+1
64.EĞER SAY<20 ISE 4. ADIMA GİT

65.TEK, CIFT YAZ
```

9. Soru

Bir n değeri için f(x)=$\sum_{x=1}^{n} \frac{1}{x^2}$ şeklindeki fonksiyonun değerini hesaplayıp ekrana yazan algoritmayı geliştiriniz.

Çözüm: Bu algoritma oldukça basit bir şekle sahip olup benzer şekildeki tüm fonksiyonları küçük değişikliklerle rahatça hesaplayabilecek bir algoritmadır. Sadece 4. adımı değiştirerek oldukça fazla sayıda algoritma veya sorun türetilebilir.

```
66.N OKU
67.F=0
68.X=1
69.F=F+1/(X*X)
70.X=X+1
71.EĞER X<=N İSE 4. ADIMA GİT

72.F YAZ
```

10. Soru

F(x)= $\sum_{x=1}^{n} \frac{1}{x^2}$ şeklindeki bir fonksiyon için f(x)>=kk şartını sağlayan en küçük n değerini bulabilecek bir algoritma geliştiriniz. Kk ve n değerlerinin klavyeden girildiğini düşünelim

Çözüm:

```
73.KK OKU
74.N OKU
75.F=0
76.X=1
```

```
77.F=F+1/(X*X)
78.X=X+1
79.EĞER F<KK İSE 5. ADIMA GİT

80.F,X YAZ
```

11. Soru

Genel Gösterimi $X_n=n^2$ şeklinde olan bir dizinin n. Elemanına kadar tüm elemanlarını ekranda yan yana gösterecek programın algoritmasını geliştiriniz.

Çözüm: Yan yana veya alt alta bu konuda algoritma düzeyinde yapabilecek bir tanım yok bu işlem daha çok Programlama dili ile hallolabilecek bir işlemdir. Bu nedenle burada o ifade dikkate alınmamıştır.

```
81.N OKU
82.X=1
83.X*X YAZ
84.X=X+1
85.EĞER X<=N İSE 3. ADIMA GİT

86.SON
```

12. Soru

Klavyeden 0-Sıfır girilinceye kadar girilen bir dizi sayının karelerinin ortalamasını bulacak program için algoritma geliştiriniz.

Çözüm: Bu algoritma 7.sorudaki algoritmaya benzer bir algoritmadır. Ve bu şekilde biri birine çok benzeyen bir sürü örnek verilebilir ve soru sorulabilir.

```
87.SAY=0
88.T=0
89.X OKU
90.EĞER X=0 İSE 8. ADIMA GİT
91.T=T+X*X
92.SAY=SAY+1
93.3. ADIMA GİT
94.ORT=T/SAY

95.ORT YAZ
```

13. Soru

Klavyeden girilecek iki pozitif tam sayının OBEB (Ortak Bölenlerin En Büyüğü)'ini bulacak algoritmayı geliştiriniz. Örneğin elimizde 3654 ve 1365 değerleri olsun. Bu değerlerin OBEB'i şu şekilde bulunmaktadır.

3654 / 1365 Kalan 924
1365 / 924 Kalan 441
924 / 441 Kalan 42
441 / 42 Kalan 21
42 / 21 Kalan 0 Kalan 0-Sıfır oluncaya kadar sıra ile bölme yapılmaktadır. OBEB değeri ise 21 olarak bulunan değerdir.

Bu işlem sırasında ve yineleme özelliklerini kullanarak iki sayının OBEB'ini bulunuz. Kalanı bulmak için % operatörünün tanımlı olduğunu varsayıp kullanabilirsiniz.

Çözüm:

```
96.A, B OKU
97.KALAN=A%B
98.EĞER KALAN=0 İSE 7. ADIMA GİT
99.A=B
100.B=KALAN
101.2. ADIMA GİT

102.B YAZ
```

14. Soru

Cos(x)=$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}\cdots$ şeklinde tanımlanmış bir Cos açılım fonksiyonu için X değeri klavyeden girilmektedir. İlk 10 terim için COS fonksiyonunun sonucunu hesaplayıp ekrana yazacak algoritmayı geliştiriniz. Bu algoritmayı geliştirirken standart işlemlerden başka fak(n), üs(taban,üs) şeklinde tanımlı fonksiyonlarını kullanabilirsiniz.

Çözüm:

```
103.X OKU
104.TERİM=1
105.KS=2
106.F=1
107.İŞARET=-1
108.F=F+İŞARET*ÜS(X,KS)/FAK(KS)
```

```
109.İŞARET=İŞARET * (-1)
110.TERİM=TERİM+1
111.KS=KS+2
112.EĞER TERİM<10 İSE 6. ADIMA GİT

113.F YAZ
```

15. Soru

Klavyeden girilen A ve B gibi iki değerin yerlerini değiştirerek ekrana yazan algoritmayı geliştiriniz.

Çözüm: Burada anlatılmak istenen herhangi bir çift değişkenin içindeki değerleri yer değiştirmektir. Bir çok programda bu tarz işlemler ihtiyaç duyulur. Örneğin birinci değerin ikinci değere göre her zaman büyük olmasını sağlayan bir algoritma söz konusu ise burada aşağıdakne benzer bir yöntem kullanılmalıdır.

```
114.A, B OKU
115.TMP=A
116.A=B
117.B=TMP

118.A, B YAZ
```

16. Soru

Klavyeden girilen bir n değerini ikilik sayı sistemine çevirip ekrana yazacak algoritmayı geliştiriniz.

Çözüm: Bu algoritma bir takım özel değişkenler ve yöntemler kullanmadan tam olarak doğru bir şekilde istenilen görevi yerine getiremeyecektir. Bu amaçla biz dizi değişkenleri kullanabileceğimizi düşünerek bu algoritmayı oluşturmaya çalışacağız.

```
119.N OKU
120.INDIS=7
121.KALAN=N%2
122.IKILI[INDIS]=KALAN
123.N=N/2
124.INDIS=INDIS-1
125.EĞER N>1 İSE 3. ADIMA GİT
126.IKILI[INDIS]=N
127.INDIS=0
128.IKILI[INDIS] YAZ
129.INDIS=INDIS+1
130.EĞER INDIS<=7 İSE 10. ADIMA GİT

131.SON
```

17. Soru

Klavyeden girilecek bir sayının tek mi çift mi olduğunu bulabilecek bir algoritma geliştiriniz. Bu algoritmayı geliştirirken başvurduğunuz varsayımları da yazınız.

Çözüm: Varsayım şu olmalıdır: Kalanı bulma operatörünün varlığı bir varsayımdır. Çünkü böyle bir operatör her programlama dilinde olmayabilir.

```
132.SAYI OKU

133.EĞER SAYI%2=0 İSE "Tek" YAZ
       DEĞİLSE "Çift" YAZ
```

18. Soru

Klavyeden girilen iki sayı ve bir operatöre göre işlem yapıp sonucu ekrana yazan algoritmayı tasarlayınız.

Çözüm:

```
134.A,B OKU
135.OP OKU
136.EĞER OP="+" İSE C=A+B
137.EĞER OP="-" İSE C=A-B
138.EĞER OP="*" İSE C=A*B
139.EĞER OP="/" İSE C=A/B

140.C YAZ
```

19. Soru

Klavyeden girilen kesirli bir değeri a/b şeklinde rasyonel ifade olarak ekrana yazabilecek algoritmayı geliştiriniz.

Çözüm: Bu algoritmayı yazabilmek için küçük bir varsayımımız olacaktır. Bu varsayıma göre programlama dilinde örneğin adı TAM olan ve bir sayısal değerin tam kısmını bulan bir fonksiyona ihtiyaç vardır.

```
141.X OKU
142.PAY=X
143.PAYDA=1
144.PAY=PAY*10
145.PAYDA=PAYDA*10
146.EĞER PAY<>TAM(PAY) İSE 4. ADIMA GİT
147.EĞER PAY%2=0 VE PAYDA%2=0 İSE
       PAY=PAY/2
       PAYDA=PAYDA/2
       7. ADIMA GİT
```

```
148.EĞER PAY%5=0 VE PAYDA%5=0 İSE
       PAY=PAY/5
       PAYDA=PAYDA/5
       8. ADIMA GİT

149.PAY, PAYDA YAZ
```

20. Soru

Klavyeden girilecek bir harfi büyük harfe çevirip ekrana yazan algoritmayı geliştiriniz.

Çözüm: Bu konu ile ilgili algoritma yazabilmek için ASCII kümesi dediğimiz bir küme hakkında bilgi sahibi olmamız gerekiyor. B küme bilgisayarınız tarafından kullanılan tüm karakterlerin saklandığı bir listedir. Normalde sayısal işlemler yapan bilgisayarınızın alfabetik karakterleri ifade edebilmesi için böyle bir lise-kümeye ihtiyaç duyulur. Bu kümenin elemanları bir çok kitapta liste olarak verilmiştir her bir liste üyesinin numarası 0 ile 255 arasındadır. Ve sayısal olarak 65 dediğimizde **"A"** harfini ve mesela 97 dediğimizde **"a"** harfini anlatmış oluruz. Bu küme incelendiğinde harfler simetrik olarak belli bir bölgede yer alırlar ve bir harfin büyüğü ile küçüğü arasında 32 gibi bir sayısal fark vardır. Büyük harfler 65 ile başlar ve daha küçük değerlere sahiptir. Küçük harfler ise 97 ile başlar.

```
150.HARF OKU
151.EĞER HARF>="a" VE HARF<="z" İSE HARF=HARF-
   32

152.HARF YAZ.
```

Bu kavramla ilgili olarak onlarca soru çözülebilir. Programlama kısmında bu konuyu temel alan bir çok örnek bulabileceksiniz.