

VERİ YAPILARI VE ALGORİTMALAR

DERS NOTU

VYA01: **Algoritma Mantığı**

Öğr. Gör. Gökhan GÜVEN

GİRİŞ	1
VERİ NEDİR?	1
Verilerin Sınıflandırılması	2
ALGORİTMA NEDİR?	2
İŞ AKIŞ ŞEMALARI	3
VERİ VE ALGORİTMA İLİŞKİSİ	5
PROGRAMLAMA DİLLERİ	6
Sabitler	7
Değişkenler	8
Değişken Tanımlama Kuralları	8
Mantıksal Sınamalar	9
Döngüler	10
ALGORİTMA ÖRNEKLERİ	10
KURT, KUZU, OT ALGORİTMASI	10
SAYI TAHMİN OYUNU	11
SAYAÇLI SAYI TAHMİN OYUNU	11
HASTALIK BELİRLEME ALGORİTMASI	11
FIBONACCI ALGORİTMASI	12

GİRİŞ

Kullanılan bilgisayar teknikleri açısından bakıldığında İşletme, İktisat ve Maliye disiplinleri diğer sosyal bilim dallarından farklı düşünülmelidir. Bu alanlarda kullanılan sayısal analizlerin birçoğu özel bilgisayar programları tarafından gerçekleştirilmektedir. Çoğu zaman bu özel bilgisayar programları analiz yapmak için yeterli gelmemekte, duruma özel yazılımlara ihtiyaç duyulmaktadır.

Ülkemizde yıllardan beridir sosyal alanlarda yapılan çalışmalarda sayısal analizlere yeterince yer verilmediği kolaylıkla görülebilir. Beşeri bilimlerde yapılan göreceli olarak az sayıdaki sayısal çalışma için ya hazır bir paket program kullanılmakta ya da programlama dillerine hâkim yazılımcıların yardımıyla bir neticeye ulaşılmaya çalışılmaktadır. Lisans düzeyinde programlamaya yeterince ya da hiç önem verilmemesi, buna bağlı olarak mezun öğrencilerin ileride yapacakları çalışmalarda sayısal tekniklerden faydalanmaması, toplumda sosyal bilimlere olan güvenin azalmasına yol açmaktadır.

Bilgisayarların metin yazma aracı olmanın çok ötesinde bir değer taşıdığının öğrencilere gösterilmesi açısından bu ders önemlidir. Yine bu ders batı toplumlarında yıllardır yapılan ekonometri, yöneylem araştırması vs. gibi alanlarda yapılan analizlerin anlaşılmasını ve geliştirilmesini de kolaylaştıracaktır. Sonuçta bu dersi alan her öğrencinin programcı olması beklenmese de, en azından kendi alanında göz dolduran çalışmalara daha çok katkı sağlayabileceği beklenebilecektir. Çünkü bilgisayar programlarının çalışma mantığını öğrenmiş bir iktisatçı, işletmeci ya da maliyeci, bilgisayarlar aracılığıyla neler yapılabileceğini daha iyi kestirebilecektir.

Türkiye'nin en büyük sorunlarından birisi ekonomi ve işletme alanında araştırmacıların güvenilebileceği ya da çalışabileceği verilerin bulunamamasıdır. Bu verilerin bulunamama sebepleri çok çeşitlidir, ancak ilk sebep belki de bu verilere olan talebin yetersiz olmasıdır. Kantitatif çalışmaların sayısının azlığı ya da sadece belli alanlara yoğunlaşmış olması niçin bu verilere olan talebin az olduğunu açıklayabilir. Sosyal bilimler alanında sayısal çalışmaların sayısını artırmanın en basit yolu lisans düzeyindeki öğrencilerin ilgisini bu tarafa çekmek olabilir. Bunu da özellikle bilgisayar destekli analiz derslerine ağırlık vererek yapmak gereklidir.

Algoritma bilgisayarlarla ilgili değildir. Nasıl müzik insan ruhuna hitap eden evrensel bir dilse, algoritma da meselelerin çözümü için kullanılan evrensel bir dildir. Algoritma düşünüş adımlarının ortaya konmasıdır. Bir algoritmanın geliştirilmesi ya da değiştirilmesi için o algoritmayı tasarlayan kişinin kültürüne, diline ya da hayat anlayışına bakmak gerekmez. Çünkü yapısı gereği algoritma insan düşüncesini ifade etmek için kullanılan bilgisayar geometrisidir.

VERİ NEDİR?

Enformasyon açıklayıcı ifadeler bütünü olarak tanımlanır. Veri ise enformasyona kaynak teşkil eden her türlü sayı, metin, renk, şekil vs. olarak tanımlanabilir. Enformasyon elde edilebilmesi için verilerin anlamlı şekilde bir araya getirilmesi gerekir. Verilerin anlamlı olarak bir araya getirilebilmesi için ilk önce sınıflamaya tabi tutulması ve daha sonra bu sınıflama içinde bir araya getirildiğinde anlamlı olacak veriler aranmalıdır. Temel olarak bilgisayarların veri işlerken yaptığı işlem budur.

Örneğin Ali, yeşil, araba gibi ifadeler veri ise, “Ali’nin arabası yeşildir” ifadesi enformasyondur. Bu verilere daha sonra 2009 yılı eklenirse enformasyon “Ali’nin yeşil arabası 2009 modeldir” şeklinde geliştirilebilir. “Hangi veri neyi ifade ediyor?” sorusu aklınıza gelmiş olabilir. İşte bu noktada verinin sınıflandırılması meselesinin önceden halledilmiş olması gerekliliğiyle karşı karşıyayız. Enformasyonu geliştirirken kullandığımız 2009 sayısı acaba bir tarihi mi, bir ölçüyü mü, yoksa bir hesaplama sonucunu mu ifade etmektedir? 2009 eğer km cinsinden verilmiş olsaydı enformasyonu “Ali’nin yeşil arabası 2009 km kullanılmıştır.” şeklinde düşünölmeyecek miydi? Daha da açarsak, eğer 2009’un tarih olduğunu biliyorsak bu neyin tarihidir? Satın alma tarihi olabilir, satış tarihi olabilir ya da bu tarih arabayı değil Ali’yi niteliyor olabilir. Yani enformasyon “Ali yeşil arabasını 2009 yılında almıştır”, “Ali yeşil arabasını 2009 yılında satmıştır” ya da “2009 doğumlu Ali’nin arabası yeşildir” şeklinde enformasyonlara ulaşma ihtimali doğar.

Örnekten de anlaşılacağı gibi verinin çok iyi sınıflandırılması doğru enformasyona ulaşılması için vazgeçilmezdir.

Verilerin Sınıflandırılması

Veri, metin, sayı, tarih, resim, video, ses veya zamandır. Bunlarda kendi içinde ayrıca sınıflandırılır. Algoritmalarda her verinin ne olduğunun önceden bildirilmesi yanlış enformasyona ulaşılmasını engelleyecektir. Çoğu üst düzey programlama dilinde her verinin ne olduğunun deklare edilmesi zorunluluktur.

Metin Veriler: Alfanümerik olarak da isimlendirilirler. Temel olarak alfabedeki harfleri içeren veri türleridir ancak bununla sınırlı değildir. Her türlü özel karakter de bunun içine dahildir. Bir veri sayı içeriyor olsa bile eğer metin olarak deklare edilmişse oradaki sayıyı sayı olarak düşünemeyiz. Çünkü sadece sayı görünümlü bir metindir. Dolayısıyla hesap yapamazsınız. Örneğin Ali verisini ikiyle çarpmanın bir anlamı yoktur, çarpamazsınız. Metin olarak deklare edilmiş 12 gibi bir ifadeyi de ikiyle çarpamazsınız. Bunun Ali ifadesinin ikiyle çarpılmasından bir farkı yoktur.

Sayı Veriler: Büyüklük ifade eden her türlü verilerdir. Kendi içerisinde başka dallara ayrılır. Bir sayı verisini tam, ondalık, karmaşık vs. gibi ayrıca sınıflamak gerekebilir.

Tarih ve Zaman Verileri: Bu tür verilerin farklı gösterimleri olabilir. Gün, ay, yıl, saat, dakika, saniye... ifade eden verilerdir.

Multimedya Veriler: Video, resim, ses vs. gibi verileri ifade etmektedir.

ALGORİTMA NEDİR?

Algoritma kelimesi matematik, astronomi ve coğrafya alanlarında çalışmış Türk düşünür Ebu Abdullah Muhammed bin Musa el-Harezmi (Abū Abdullāh Muhammad ibn Mūsā al-Khwārizmī) Latince (Algoritmi) isminden türetilmiştir. IX. yüzyılda Bağdat’ta yaşamış olan El-Harezmi dijital elektroniğin temelini oluşturan ikilik sayma sistemini (binary) ve sıfırı bulmuştur.

Algoritma kısaca çözüm adımlarıdır. Farkında olmasak da hayatın her anında algoritma kullanırız. Evden okula ulaşmak için ulaşım algoritmasını, açlığımızı gidermek için yemek algoritmasını tercih ederiz. Amaç okula ulaşmak ya da doymak ise algoritmadaki bir hata oldukça can sıkıcı olabilir. Okula ulaşma algoritmasını ele alalım. Her algoritmada “Başla” ve “Bitir” ifadelerine yer verilir.

1. Başla
2. Otobüs durağına git.
3. Üniversiteye giden otobüsü bekle.
4. Otobüse bin.
5. Biletini okut.
6. Üniversite durağında in.
7. Okula gir.
8. Bitir

Basit gibi görünen bu algoritma üzerinde bir hata yapalım. Örneğin 3 numaralı adımı atlayalım. Eğer yeterince şanslı isek doğru sonuca ulaşabiliriz (Yani üniversiteye giden otobüse binebiliriz). Ama büyük ihtimalle ilk gelen otobüse bindiğimiz için şehrin üniversiteden uzak bir semtine gideceğimizden sonuncu adımı asla gerçekleştiremeyeceğiz. Hergün yaptığımız ama yaparken düşünmediğimiz birkaç eylem için algoritmayı yukarıdaki gibi kurmak algoritma kurma prensiplerini anlamayı kolaylaştıracaktır.

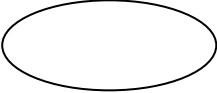

Algoritma kurulurken dikkat edilecek hususlar şunlardır;


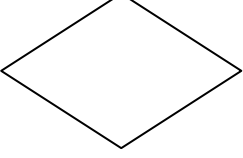
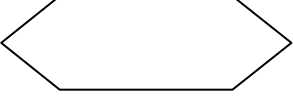
1. Mutlaka bir başlangıcı ve sonu olmalıdır. Sonu olmayan bir algoritmanın test edilmesinin mümkün olmadığı açıktır.
2. Bir algoritmadaki ifadeler kesin olmalıdır. Belirsizlikler algoritma ile ulaşmak istenen sonucun elde edilememesine neden olur.
3. Algoritma sınırlı sayıda belirli kurallarla kurulmalıdır. Ve bu belirli kurallar takip edecekleri bir sıraya sahip olmalıdırlar.

İŞ AKIŞ ŞEMALARI

İş akış şemaları kurulan algoritmalar üzerinde çalışmayı kolaylaştıran yapılardır. Aynı zamanda algoritmanın başka geliştiriciler tarafından kolaylıkla anlaşılmasını da sağlarlar. Kurulan bir algoritmanın iş akış şemasına aktarılması oldukça kolay bir iştir. İş akış şeması ne kadar ayrıntılıysa kurulan algoritma üzerindeki hataların giderilmesi ve geliştirilmesi de o derece kolaydır.

İş akış şemalarında algoritmadaki her işlemi göstermek için farklı şekiller kullanılır. Genel olarak kullanılan şekillerden birkaçı şunlardır;

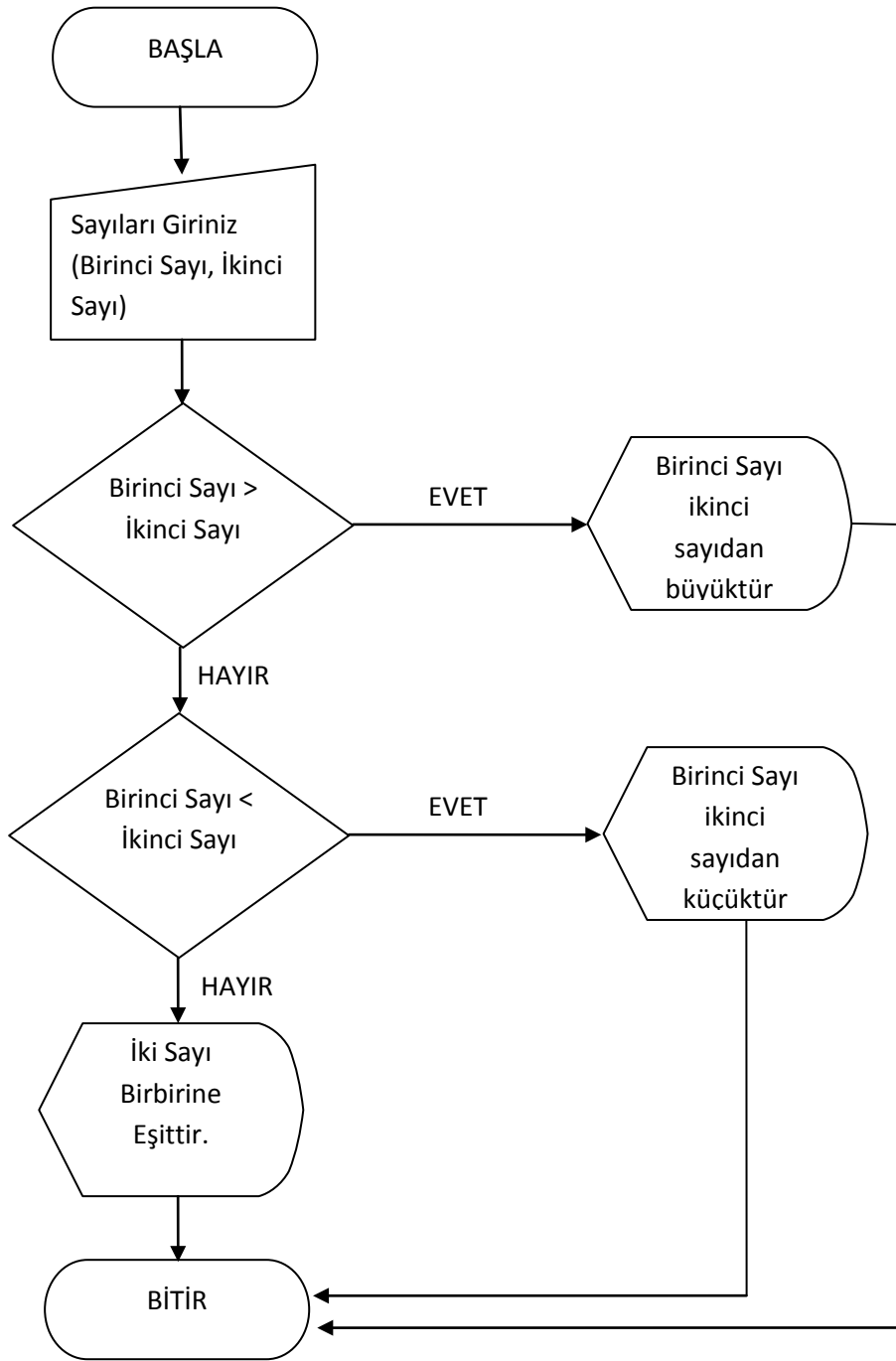
	Elips: Başla ve Bitir ifadeleri için kullanılır.
	Paralel Kenar: Algoritmadaki veri girişi ve işlemlerden sonra elde çıktıları gösterir.(Oku, Yaz)

	Dikdörtgen Değer atama ve matematiksel işlemleri göstermek için kullanılır. ($x=x-y$, fiyat=1500 vs..)
	Eşkenar Dörtgen Mantıksal sınaıma ifadelerini göstermek için kullanılır. (Eğer sonuc<50 ise Yaz "Başarısız")
	Altıgen Algoritma içinde tekrarlanan eylemleri göstermek için kullanılır. (Döngüleri gösterir.)

Başlarda iş akış şemasını çizmeden önce mutlaka algoritmanın ortaya konması gerekir. İş akış şemasını yazmada ustalaştıkça algoritmanın ayrıca yazılmasına gerek kalmayabilir.

Algoritma ve İş Akış Şeması

1. Başla
2. Birinci sayıyı gir.(x), İkinci sayıyı gir.(y)
4. Eğer $x > y$ ise ekrana Birinci sayı büyüktür yaz.
5. Eğer $x < y$ ise ekrana Birinci sayı küçüktür yaz.
6. Ekrana sayılar birbirine eşittir yaz.
7. Bitir



VERİ VE ALGORİTMA İLİŞKİSİ

Ortada bir sorun ya da amaç yoksa insanın düşünmesine de gerek yoktur. Bu iddialı ifade algoritmalar için de geçerlidir. Bir sorunu çözmek ya da herhangi bir amaca ulaşmak için algoritma kurarız. Veri algoritmanın hammaddesidir ve algoritmanın girdisidir. Çözüm ya da beklenen amaç ise algoritmanın çıktısıdır. Algoritmada iş akışı sürecine sokulan her veri mutlaka bir işlemin tetikleyicisi olur. Veri algoritmaya ya elle girilir ya da önceden dijital ortama girilmiş verilerden faydalanılır. Çoğu zaman birkaç veri çeşitli işlemlerden sonra biraraya getirilerek süreç boyunca kullanılacak yeni bir veri elde edilir.

Algoritma içindeki verilerin hangi türe ait olduğunun en başta bildirilmesi iş akış sürecinin ilerdeki safhalarında karışıklıkların ortadan kaldırılmasına yardımcı olacaktır. Temel matematik kuralı algoritmalar için de geçerlidir. Sadece aynı tür veriler ile işlem yapılabilir. Bazı özel durumlarda farklı türler birbirine çevrilebilir. Hangi tür veri türü ile ilgili işlem yapılmışsa sonuç aynı türdür. Yani iki metin üzerinde işlem yapılmışsa sonuç yine metindir. İki zaman üzerinde işlem yapılmışsa sonuç zamandır. İki sayı üzerinde işlem yapılmışsa sonuç sayıdır vs..

PROGRAMLAMA DİLLERİ

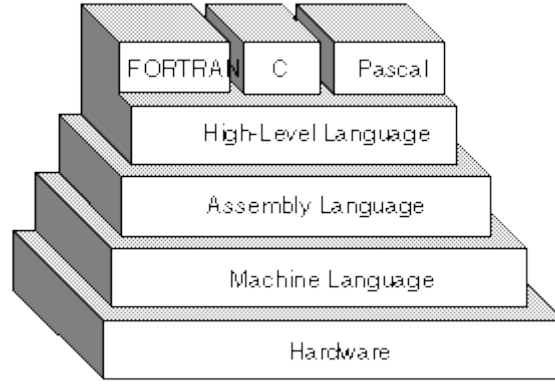
Bilgisayarların temel bileşeni olan CPU'nun yapıtaşı transistördür. Transistörün elektronik devre içindeki görevi ise devreyi açıp kapamaktır. Dolayısıyla devre açıkken 0, devre kapalıyken 1 değerini alırlar. Basitçe bir CPU içinde bulunan milyonlarca hatta milyarlarca transistörün hangisinin açık hangisinin kapalı olması gerektiğini belirlemeye programlama diyoruz. Bu açıklamadan da anlaşılacağı gibi bilgisayar 0 ve 1 lerden ibarettir ifadesinin dayandığı prensip budur.

Eğer bilgisayar programlamadan bahsediyorsak programlama dillerine değiniyoruz demektir. Temel sınıflamada iki çeşit programla dili vardır; alt düzey (low-level), üst düzey (high-level) programlama dilleri.

Bilgisayarların doğrudan anlayabildiği tek dil makine dilidir. Üst düzey programlama dillerinde yazılan programlar özel derleyicilerle makine diline çevrilerek çalıştırılırlar. Makine dili, sadece 0 ve 1'lerden oluşur ve bu dilde program yazmak oldukça zaman alıcı ve karmaşık bir iştir.

Assembly dili makine dilinden daha kolay olmasına karşın yine de yazması zor ve sıkıcıdır. İngilizceden türetilmiş kısaltmaların kullanıldığı bu dil düşük düzey programlama dilidir. Assemblers adı verilen çevirici yazılımlar yardımıyla assembly dilinde yazılan programlar makine diline çevrilir. Üst düzey programlama yapan uzman yazılımcılar assembly'e hâkim olunması gerektiğini bilir.

Üst düzey programlama dilleri belirli bir gramer ve söz dizim kuralları kullanılarak program yazmaya yarayan ara yüzlerdir. Üst düzey programlama dillerinde yazılan programların tek başına çalışabilmesi için derleyiciler (compiler) yardımıyla makine diline çevrilmesi gerekir. Üst düzey programlama dillerinin alt düzey programlama dillerine göre öne çıkan avantajları anlamanın, yazmanın ve yeniden düzenlemenin çok daha kolay olmasıdır. Düzinelerce üst düzey programlama dili sayılabilir. Bunlardan en ünlüleri ise şunlardır; BASIC, COBOL, C, C++, FORTRAN, LISP, Pascal, Prolog, Java. İşletim sistemlerinin GUI (Graphical User Interface) desteğini vermesi, programlama dillerinin de görsel olarak zenginleşmesini sağlamıştır. Artık yazılım geliştiriciler görsel elementleri kullanarak program geliştirir olmuşlardır. Günümüzde birçok programlama dilini aynı anda destekleyen editörler yardımıyla program yazmamızı sağlayan editör programları kullanılmaktadır.



Hangi düzey programlama dili olursa olsun tamamı algoritma mantığını aynen kullanır. Değişen sadece yazım kurallarıdır. Algoritmalar göz önüne alındığında bir programlama dilinin bir diğerine üstünlüğü söz konusu değildir. Ancak üst düzey programlama dilleri arasında üstünlük ayrımı yapanlar, yazım kurallarının basitliğine, parametre sayısına ve derleyici hızına bakarak yorum yapmaktadırlar.

Algoritma kurmayı daha basit hale getirmek için kendi başınıza bir dil oluşturabilirsiniz. Nihayetinde algoritma programlama dillerinden bağımsızdır. Algoritma kurma mantığını anladığınızda geriye sadece kurduğunuz algoritmayı yazım kurallarına göre ilgili programlama diline geçirmek kalacaktır.

Şimdi üç sayının ortalamasını bulduran bir algoritmayı, kendi uydurduğumuz bir programlama diliyle yazalım;

1. Başla
2. Gir; Birinci Sayı
3. Gir; İkinci Sayı
4. Gir; Üçüncü Sayı
5. Hesapla ve Ortalama Diye Adlandır $\{(Birinci\ Sayı + İkinci\ Sayı + Üçüncü\ Sayı) / 3\}$
6. Ekrana Ortalama değerini yaz
7. Bitir

Programlamayı yeni öğrenenler herhangi bir programlama dili ile başlaması gerektiğini düşünür. Ama programlamaya başlamanın en doğru yolu algoritma öğrenmektir. İyi algoritma, çözümü en kestirme yoldan bulan algoritmadır. Kurulan algoritmanın çalışması her zaman yeterli olmayabilir. Unutulmamalıdır ki kurulan algoritma kısıtlı kaynakları bulunan bir cihaz üzerinde çalıştırılacaktır.

Sabitler

Algoritma içindeki her tür alfabetik, sayısal ya da zamansal değerlere sabit denir. Alfabetik sabitlere alfasayısal sabit de denir. Alfasayısal sabitler tırnak içinde gösterilirler. Alfasayısal sabitlerle aritmetik işlem yapılamaz. Bir sayı algoritma içinde tırnaklar içinde gösterilmişse artık sayı niteliği taşımaz. Sadece sayıya benzeyen alfasayısal bir değerdir ve aritmetik işlem içinde kullanılamaz. Her sabit algoritmaya dahil edeceğimiz bir veridir. Buradan her sabitin algoritma için bir girdi olduğu sonucuna ulaşmanız yanlış olmayacaktır.

Değişkenler

En basit tanımıyla değişken, bir değerin saklanması için hafızada ayrılan yerdir. Daha açık ifadeyle bir değeri gösteren semboldür. Değişken kullanımı matematikle aynıdır. Bir değişkenin değeri hangi tür veriyi içeriyorsa değişkende o tür bir değişkendir. Yani metin içeren bir değişken için metin değişken, sayı içeren bir değişken için sayı değişken, tarih ya da zaman içeren bir değişken içinse tarih ya da zaman değişken isimleri kullanılır.

Değişken Tanımlama Kuralları

Aslında kendi oluşturduğumuz dilde böyle kurallara ihtiyaç yoktur. Ancak kurulacak algoritmanın bir şekilde programa dönüştürülmesi gerekir. Alışkanlık haline gelmesi bakımından kendi uyduracağımız dil için yaşayan programlama dillerinde kullanılan değişken tanımlama kurallarını kullanmak faydalı olacaktır. Değişken tanımlama kuralları şunlardır;

1. Değişkenler isimlendirilirken rakam ile başlanmaz. Değişken isminin ilk karakteri mutlaka bir harf olmalıdır. Daha sonra rakamlar kullanılabilir.
2. Değişkenler İngiliz alfabesinin karakterleri kullanılarak adlandırılırlar. Türkçe alfabede yer alan ç, Ç, Ö, ö, Ş, ş, Ğ, ğ, ü, Ü, İ, ı harfleri değişken ismi içinde yer alamazlar.
3. Değişken isimlerinde boşluk ve :, *, /, +, },], ! vs. gibi özel karakterler bulunmaz.

Üç sayının ortalamasını almak için kullandığımız algoritmada bu tür kurallara dikkat edilmemişti. Şimdi açıkladığımız bu kurallara göre algoritmamızı yeniden yazalım.

1. Başla
2. Gir; BirinciSayı
3. Gir; İkinciSayı
4. Giri; UcuncuSayı
5. Hesapla ve Ortalama Diye Adlandır $\{(BirinciSayı + İkinciSayı + UcuncuSayı) / 3\}$
6. Ekrana Ortalama değerini yaz
7. Bitir

Algoritmayı gerçekleştirmek için kullanacağımız programlama diline göre kurallarda ufak tefek değişiklikler olabilir. Ancak üst düzey programlama dillerinin genelinde yukarıda bahsettiğimiz değişken tanımlama kurallarına uymanız gereklidir. Algoritmada kurma mantığı gereği belirsizlikten özellikle kaçınılır. Bundan dolayı algoritma içinde kullanılacak değişkenler daha adımların en başında türleriyle birlikte belirtilir (deklare edilir). Üç sayının ortalamasını alan algoritmamızı bu kurala uygun hale getirelim.

1. Başla
2. *BirinciSayı, İkinciSayı, UcuncuSayı, Ortalama sayı içerecek değişkenlerdir.*
3. Gir; BirinciSayı
4. Gir; İkinciSayı
5. Giri; UcuncuSayı
6. Hesapla ve Ortalama Diye Adlandır $\{(BirinciSayı + İkinciSayı + UcuncuSayı) / 3\}$
7. Ekrana Ortalama değerini yaz
8. Bitir

Bir değişken içine mutlaka bir sabit yerleştirilir. Bu işe atama işlemi denir ve genellikle “=” işareti kullanılarak gerçekleştirilen bir işlemdir. Örneğin renk değişkenine, yeşil ifadesini atamak istiyorsak; renk=”yeşil” yazmamız genellikle yeterlidir.

Algoritmamızı bu yeni kuralara göre yeniden düzenleyelim;

1. Başla
2. BirinciSayi, İkinciSayi, UcuncuSayi, Ortalama sayı içerecek değişkenlerdir.
3. Gir; BirinciSayi
4. Gir; İkinciSayi
5. Giri; UcuncuSayi
6. $Ortalama = \{(BirinciSayi + İkinciSayi + UcuncuSayi) / 3\}$
7. Ekrana Ortalama değerini yaz
8. Bitir

Mantıksal Sınamalar

Algoritmanın temelini mantıksal sınamalar oluşturur. Mantıksal sınamalar değişkenler arasında büyüklük, küçüklük, eşitlik sınaması yapmak için kullanılır. Sınamanın sonucu ya doğrudur ya da yanlıştır. İş akış şemasında sınama sonucu doğruysa Evet okunun gösterdiği yöne, yanlışsa Hayır okunun gösterdiği yöne doğru ilerlenir. Algoritmamıza bir mantıksal sınamalar yerleştirelim.

1. Başla
2. BirinciSayi, İkinciSayi, UcuncuSayi, Ortalama sayı içerecek değişkenlerdir.
3. Gir; BirinciSayi
4. Gir; İkinciSayi
5. Giri; UcuncuSayi
6. $Ortalama = \{(BirinciSayi + İkinciSayi + UcuncuSayi) / 3\}$
7. *Eğer Ortalama değeri 1’den büyükse ekrana “sayıların toplamı 3’den büyük” değilse “Sayıların toplamı 3’den küçük” yaz.*
8. *Eğer Ortalama değeri 1’e eşitse “Sayıların toplamı 3’tür”*
9. Ekrana Ortalama değerini yaz
10. Bitir

Algoritmamızı basitleştirmek için matematikte kullanılan çeşitli operatörlerden faydalanmak doğru olacaktır.

> işareti sağdakinin soldakinden büyük olup olmadığını sınamak için kullanılır.

< işareti sağdakinin soldakinden küçük olup olmadığını sınamak için kullanılır.

= işareti eşitlik durumu olup olmadığını kontrol etmek için kullanılır. Eşittir işaretinin aynı zamanda bir değişkene değer atamak gibi bir görevi vardır. Mantıksal sınama içinde ise eşitlik testi yapmak içindir.

>= Büyük eşittir.

<= Küçük eşittir.

<> Eşit değildir.

Algoritmamızı yeniden düzenleyelim.

1. Başla
2. BirinciSayi, İkinciSayi, UcuncuSayi, Ortalama sayı içerecek değişkenlerdir.
3. Gir; BirinciSayi
4. Gir; İkinciSayi
5. Giri; UcuncuSayi
6. $Ortalama = \{(BirinciSayi + İkinciSayi + UcuncuSayi) / 3\}$
7. *Eğer Ortalama > 1 ise ekrana “sayıların toplamı 3’den büyük” değilse “Sayıların toplamı 3’den küçük” yaz.*
8. *Eğer Ortalama = 1 ise “Sayıların toplamı 3’tür”*
9. Ekrana Ortalama değerini yaz
10. Bitir

Yukarıdaki algoritmanın 6. Satırında eşittir işareti ortalama değişkenine değer atamak için kullanılmıştır. 8. Satırda ise mantıksal sınamanın içinde kullanılan = işaretinin eşitlik testi yapmak için kullanıldığına dikkat ediniz. İkisi de aynı işaret gibi görünse de amaçları tamamen farklıdır.

Döngüler

Algoritma içindeki bir adımın ya da adımlar grubunun istenilen sayıda tekrar tekrar yapılmasını sağlayan yapılardır. Kullanımlarına dikkat edilmelidir. Eğer bir şekilde döngüden çıkılamıyorsa bu büyük bir hatadır ve kurulan algoritma asla sonuç vermez. Bu duruma da kısır döngü denir. İşlemlerin kesin olarak belirli bir sayıda yapılması isteniyorsa döngü içinde sayaç değişken adını verdiğimiz değişkenler kullanılır. Bu değişkenler döngü içerisinde mutlaka değişikliğe maruz kalırlar. Eğer sayaç değişkenlerde bir değişiklik olmazsa döngü kısır döngü haline gelir.

ALGORİTMA ÖRNEKLERİ

KURT, KUZU, OT ALGORİTMASI

Sizden kurt, kuzu ve ot’u, sadece bir tanesini yanınıza alabileceğiniz bir sandalla nehrin öbür yakasına geçirmeniz isteniyor. Siz olmadan kurtla kuzu yan yana kalamıyor kuzu ile de ot yan yana kalamıyor. Dolayısıyla kuzu ile otu, kurt ile kuzuyu tek başlarına nehrin herhangi bir yakasında bırakamazsınız. İlgili algoritmayı yazalım.

1. Başla
2. Kuzuyu al karşıya geçir.
3. Nehrin diğer yakasına dön.
4. Kurt’u al karşıya geçir.
5. Kuzuyu yanına al nehrin diğer yakasına dön.
6. Kuzuyu bırak otu al nehrin karşı yakasına geçir.
7. Nehrin diğer yakasına dön.
8. Kuzuyu al karşıya geçir.
9. Bitir.

SAYI TAHMİN OYUNU

1. Başla
2. tahmin, z değişkenleri tam sayı içeren değişkenlerdir.
3. z değişkeni için 1 ile 100 arasında rastgele bir sayı oluştur (Bilgisayar tarafından yapılıyor)
4. Gir; tahmin
5. Eğer $z > \text{tahmin}$ ise ekrana "Tuttuğum sayı daha büyük" yaz ve dördüncü adıma git.
6. Eğer $z < \text{tahmin}$ ise ekrana "Tuttuğum sayı daha küçük" yaz ve dördüncü adıma git.
7. Ekrana "Tebrikler tuttuğum sayı buydu" yaz.
8. Bitir

SAYAÇLI SAYI TAHMİN OYUNU

1. Başla
2. tahmin, z, sayac tamsayı içeren değişkenlerdir.
3. Kafadan z değişkeni için 1 ile 100 arasında bir sayı tut
4. Sayac değişkeninin değerini 0 olarak belirle.
5. Gir; tahmin
6. Sayac değişkeninin değerini 1 artır.
7. Eğer $z > \text{tahmin}$ ise ekrana "Tuttuğum sayı daha büyük" yaz ve beşinci adıma git.
8. Eğer $z < \text{tahmin}$ ise ekrana "Tuttuğum sayı daha küçük" yaz ve beşinci adıma git.
9. Ekrana "Tebrikler tuttuğum sayı buydu" yaz. Sayac değişkeninin değerini ekrana yaz ve peşine " denemede bildiniz" yazısını ekle.
10. Bitir

HASTALIK BELİRLEME ALGORİTMASI

Aşağıda bazı hastalıkların belirtileri verilmiştir. Bu belirtilerden faydalanarak hangi hastalığa kişinin yakalandığını bulan algoritmayı kurunuz.

Verem; Öksürük, Yüksek Ateş, Halsizlik, Yorgunluk, kusma.

Nezle; Öksürük, Halsizlik

Tansiyon; Baş dönmesi, Yıldızlar görüyor musunuz?

Zehirlenme; Kusma, Halsizlik.

1. Başla
2. oksuruk, yuksekates, halsizlik, yorgunluk, kusma, basdonmesi, yildiz, hastalik metin değişkenlerdir.
3. Ekrana "Öksürük var mı? (E/H)" yaz, Gir; oksuruk
4. Ekrana "Yüksek ateş var mı? (E/H)" yaz, Gir; yuksekates
5. Ekrana "Halsizlik var mı? (E/H)" yaz, Gir; halsizlik
6. Ekrana "Yorgunluk var mı? (E/H)" yaz, Gir; yorgunluk

7. Ekranı "Kusma var mı? (E/H)" yaz, Gir; kusma
8. Ekranı "Baş dönmesi var mı? (E/H)" yaz, Gir; basdonmesi
9. Ekranı "Yıldızlar görüyor musunuz? (E/H)" yaz, Gir; yıldız
10. Eğer oksuruk ="E" ve halsizlik="E" ve yuksekates<>"E" ve yorgunluk<>"E" ve kusma<>"E" ise hastalık="Nezle"
11. Eğer kusma="E" ve halsizlik="E" yuksekates<>"E" ve yorgunluk<>"E" ve oksuruk<>"E" ise hastalık="Zehirlenme"
12. Eğer oksuruk = "E" ve yuksekates ="E" ve halsizlik="E" ve yorgunluk="E" ve kusma="E" ise hastalık="Verem"
13. Eğer basdonmesi="E" ve yıldızlar="E" ve kusma<>"E" ve halsizlik<>"E" yuksekates<>"E" ve yorgunluk<>"E" ve oksuruk<>"E" ise hastalık="Tansiyon"
14. Eğer basdonmesi<>"E" ve yıldızlar<>"E" ve kusma<>"E" ve halsizlik<>"E" yuksekates<>"E" ve yorgunluk<>"E" ve oksuruk<>"E" ise hastalık="Teşhis konulamadı"
15. Ekranı hastalık değerini yaz.
16. Bitir.

FIBONACCI ALGORİTMASI

Fibonacci dizisi sayıları: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, ... şeklinde devam eder. Her sayı kendisinden önce gelen iki sayının toplamıdır. Klavyeden girilen n'ci fibonacci sayısını bulmak için bir algoritma kuralım.

1. Başla
2. a, ad, sayac, n, x tam sayılı değişkenlerdir.
3. a=1, ad=0, sayac=0
4. Gir; n
5. x=a+ad
6. ad=x
7. a=x-a
8. sayac=sayac+1
9. Eğer sayac<>n ise 5. Satıra git
10. Ekranı x değerini yaz.
11. Bitir