ÜNİTE 01, Algoritma Geliştirme, Öğr. Gör. Erkan HÜRNALI

ALT BAŞLIKLAR

1. Algoritma Geliştirme
   1.1. Algoritma Nedir?
   1.2. Problem Nedir?
   1.3. Algoritma Adımları
   1.4. Algoritmaların Taşıması Gereken Özellikler
   1.5. Bilgisayar Programı Nedir?
   1.6. Algoritma Örnekleri
   1.7. Algoritma Çalıştırma

ÜNİTE HAKKINDA

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. Birçok 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 kâğıt üzerinde gerçekleştirilir ve elde edilen verilere göre program yazılır.

Ancak programlamaya yeni iseniz hemen program yazmanız oldukça zor olacaktır. Öncelikle programlama mantığını ve problem çözme önsezisini edinmeniz gerekecek. Bu ünite sizlerin bu sezgiyi kazanmanızı ve bu mantığı oluşturmanızı sağlayacak ön bilgiler sunmaktadır.

Ne var ki bahsettiğimiz mantık ve sezgiler program yazdıkça gelişir. Birçok kişi tarafından programlama yeteneğinin doğuştan olduğu söylense bile (ki gerçekte doğruluk payı yüksektir) bu yeteneği edinmek resim yapma yeteneğini edinmekten ya da güzel sanatlara karşı bir yeteneği edinmekten çok daha kolaydır. (by., 2007)

ÖĞRENME HEDEFLERİ

Bu ünite bitiminde;

- Algoritma nedir? Tanımlayabilecek,
- Problem nedir? Tanımlayabilecek,
- Çevrenizdeki problemlerin farkında olabilecek,
- Problemlerin çözüm aşamalarını içeren algoritma adımlarını oluşturabilecek ve
- Algoritma adımlarının taşıması gereken özellikleri bileceksiniz.

ÜNİTEYİ ÇALIŞIRKEN

Algoritma Geliştirme Ünitesi, daha çok sözel kavramlardan oluştuğu için ve ayrıca yoruma da çok açık bir konu olduğu için; konuyu değişik kaynaklardan da taradıktan sonra mümkünse kendi aranızda tartışarak bu üniteyi daha iyi kavrayabilirsiniz.

Ayrıca ünitede yeri geldikçe, günlük hayatımızdaki örneklere algoritma geliştirme açısından yaklaşılmaya çalışılmıştır. Sizde bu konudaki farkındalığınızı arttırmak için etrafınızdaki örnekleri bir fırsat bilip, daha çok algoritma geliştirme gözlüğüyle bakmayı deneyebilirsiniz.

ANA METİN

1. Algoritma Geliştirme

1.1. Algoritma Nedir?

✓ Algoritma en yalın tanımıyla bir problemin çözüm aşamalarıdır. Diğer bir ifadeyle problemi çözmek için tespit edilen işlem basamaklarıdır.

Dolayısıyla çözülmek istenen bir problemin öncelikli olarak işlem basamaklarının belirlenmesi yani algoritmasının kurulması gerekmektedir.

* Örneğin çok basit bir problemi ele alalım: Problemimiz, "Okula varmak" olsun. Bu problemi çözmek için gerekli olan işlem basamakları nelerdir birlikte bakalım.

- Uyanmak (Uyanmadan okula gitmemeliyiz. Gidebilir miyiz? Evet, gidebiliriz fakat bu istenmeyen bir durumdur.)
- Elimizi yüzümüzü yıkamak. (Aynı şekilde elimizi yüzümüzü yıkamadan da okula gidebiliriz fakat bu da istenmeyen bir durumdur.)
- Kahvaltı yapmak (Yine aynı şekilde kahvaltı günün en önemli öğünlerindendir…)
- Hazırlık (…)
- Ulaşım araçlarını kullanmak (…)
- Okula varış (…ve problem çözülmüş oldu)

Bir problemi çözmek için öncelikli olarak algoritmasının kurulması gerektiğinden bahsetmiştik. Yukarıdaki örneğimizde de basit bir problemin algoritmasını kurmuş olduk. Yani bu problemi çözebilmek için gerekli olan işlem basamaklarını belirledik. Dikkat ederseniz bu işlem basamakları kesin değildir. Kişiden kişiye göre değişebilir. Örneğin elimizi yüzümüzü yıkamadan veya kahvaltı yapmadan da okula varabiliriz. Ya da kahvaltı yaptıktan sonra da elimizi yüzümüzü yıkayabiliriz. Bu, programcı olarak sizin probleme nasıl yaklaştığınızla ilgilidir.

! Uzun veya kısa, bir şekilde, problemin algoritması hazırlanabiliyor ise o problem çözülebilir demektir.

1.2. Problem Nedir?

Aslına bakılırsa şu ana kadar bahsedilen problem ve çözümü tamamen konuyu açıklamak amacıyla kasten seçilmiş çok da mantıklı olmayan basit bir örnektir.

Gerçek hayatta problem, bir işlemin, otomasyonun ya da bilimsel hesaplamanın bilgisayarla çözülmesi fikrinin ortaya çıkmasıdır. Bu tip fikirlerde insanların bu sorunları beyinle çözmeleri ya imkânsızdır 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.

1.3. Algoritma Adımları

! İşlem basamakları, gerekli kontrol ifadeleri ile geliştirilerek problemin çözümü sağlamlaştırılabilir ve çözüm daha kararlı bir yapıya getirilebilir.

Yine yukarıdaki basit örnekte, problemimiz okula varmaktı ve ilk aşamayı uyanmak olarak belirlemiştik. Uyanmadan da bu aşama geçilebilir mi? Eğer ki algoritmamızı bu şekilde bırakırsak, evet, geçilebilir ve biz okula uyanmadan yani uyur-gezer bir şekilde varmış oluruz. Bu da çözümümüzün kararsız sonuçlar üretmesine sebep olur. İşletim sistemlerinde durduk yerde çıkan mavi ekranları hatırlayın, işlem basamaklarının kararsızlığından dolayı çözüm bir yerde çökmüştür.

İşte bu noktada çözüm aşamalarından bazılarının (daha doğrusu gerekli olanlarının) sağlamlaştırılması gerektiği, yani daha karalı bir yapıya kavuşturulmaları gerektiği ortaya çıkmaktadır.

Bunun için problemimizin çözümündeki ilk aşamayı;

- Uyan
- Uyanıp-uyanmadığını kontrol et
- Eğer uyanmadıysan başa dön, uyandıysan devam et…

şeklinde değiştirirsek, yani sonrasında gelen aşamada uyanıp-uyanılmadığını kontrol ettirirsek, uyanmadan okula gitmemeyi garantilemiş oluruz.

Yaptığımız bu kontrolde, istediğimiz şartların oluşmaması durumunda çözümü, problemin başına yönlendirmekteyiz. Peki ya geliştirdiğimiz algoritmaya bağlı olarak çözümü problemin ortalarında herhangi bir satıra yönlendirmek isteseydik; bunu yapabilmek için çözümümüzü oluşturan işlem basamaklarını numaralandırmamız gerekecekti.

✓ Programcılık dilinde numaralandırılmış olan bu işlem basamaklarının her birine bir “algoritma adımı” denmektedir.

Yukarıdaki aşamayı algoritma adımları ile ifade edecek olursak aşağıdaki gibi olur:

1. Uyan
2. Uyanıp-uyanmadığını kontrol et
3. Eğer uyanmadıysan 1. Adıma geri dön, uyandıysan 4.Adımdan devam et
4. Elini-yüzünü yıka…

! Algoritma adımlarının numaralandırılmasının sebebi, herhangi bir adımda ilgili adımı tarif edebilmek içindir. (1. adıma geri dön, 9 adıma atla gibi…)

1.4. Algoritmaların Taşıması Gereken Özellikler

Algoritmalar günlük konuşma diline yakın bir dille yazılabileceği için fazlaca formal değildir. Yine de bir algoritma için aşağıdaki ifadelerin mutlaka doğrulanması gereklidir.

- Her adım son derece belirleyici olmalıdır ve aynı zamanda hiç bir şey şansa bağlı olmamalıdır (yukarıdaki örnekte uyanmak adımının geliştirilerek şansa bırakılmadığı gibi)
- Belirli bir sayıda adım sonunda algoritma sonlanmalıdır (sonsuz bir döngüye girmemelidir)
- Algoritmalar karşılaşılabilecek tüm ihtimalleri ele alabilecek kadar genel olmalıdır.

1.5. Bilgisayar Programı Nedir?

Bir bilgisayar programı aslında sıra düzensel olarak tanımlanmış bir dizi algoritma adımından başka bir şey değildir. Bu açıdan bizim yazmaya çalışacağımız programda bir dizi algoritma adımı 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 biraz daha geliştirerek 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.

1.6. Algoritma Örnekleri

Şimdide başka basit problemlerin çözümlerini (işlem basamaklarını) algoritma adımları şeklinde yazmaya çalışalım.

* Örnek: 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. Pişmiş mi 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 soğumasını bekle
19. Keki servis edebilirsin.

Bu algoritma günlük hayattan bir örnek. Gerçekte biz her işimizi algoritmik olarak yaparız ancak bunun 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.

Algoritmada algoritmanın genel işleyişini etkileyebilecek hiç bir belirsizlik olmamalıdır. (Bu örnekte öyle bir belirsizlik var. Bir fırının yeteri kadar ısınabilmesi hangi koşula bağlıdır, bu fırın ne zaman açılmış olmalıdır ve kaç dereceye ayarlanmış olmalıdır gibi...) Algoritmada bazı adımlar yer değiştirebilir. Ancak birçok 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. (by., 2007)

? Peki, bilgisayarda çözülecek bir sorunu nasıl bir algoritma ile ifade ederiz?

Bunun için öncelikle bir sorun tanımlayalım ve başlangıçta 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. Sayıları oku
2. Sayıların toplamlarını hesapla
3. Toplamlarını ekrana yaz

Yukarıdaki algoritma adımları problemi çözmek için yeterlidir. Yalnız işlem basamaklarını bir bilgisayara uygulatmaya çalıştığımızda algoritmaların taşıması gereken özelliklerden birincisiyle çelişmiş oluruz.

Çünkü her bir adım son derece belirleyici değildir. En azından bilgisayarlar için. Zira biz insanlar için bu algoritma adımları gayet belirleyicidir. Fakat gelin görün ki problemleri çözmek için hazırladığımız ve mantıksal bir sıraya dizdiğimiz bu işlem basamakları bilgisayarlar tarafından yürütülecektir, insanlar tarafından değil! (Neden, çünkü bilgisayarlar bizden çok daha hızlı işlem yapabilmektedirler ve hafızaları da bizden çok daha kuvvetlidir).

Onun için işlem basamaklarımızı oluştururken bilgisayarların bu işlem basamaklarını oluşturan algoritma adımlarını anlayabilmeleri ve yürütebilmeleri için bilgisayar diline yakın bir dille oluşturulmasına dikkat etmemiz gerekmektedir.

Öyle ki, neredeyse tüm çözümlerimizde algoritmamızı BAŞLA ifadesiyle başlatıp DUR ifadesiyle de durdurmamız gerekmektedir:

1. Başla
2. Sayıları oku
3. Sayıların toplamlarını hesapla
4. Toplamlarını ekrana yaz
5. Dur

şeklinde… Fakat yeterli değildir. Çünkü hala belirsizlikler devam etmektedir:

"Sayıları oku ifadesi" bilgisayar için hala muallâktadır (bilinmeyenlerle doludur). Tamam, sayılar okunacaktır ama hangi sayılar ve nereye okunacaktır?

! Oku ifadesi bilgisayar dilinde genellikle, kullanıcının konsoldan yaptığı bilgi girişlerini ifade etmektedir ve girilen bu bilginin diğer algoritma adımları tarafından da işlenebilmesi için ilgili algoritma adımına kadar taşınması gerekmektedir.

Bilgisayarlarda bu taşıma işlemi ileriki konularda daha da detaylı işleyeceğimiz değişkenler vasıtasıyla yapılmaktadır.

✓ Değişkenleri şimdilik içerisinde verileri tutabileceğimiz boş kutucuklar (bellek alanları) olarak düşünebilirsiniz.

Burada da sayılarla sonraki algoritma adımlarında işlem yapmamız gerektiğinden onları birer değişken içerisine okutmamız (koymamız) gerekir.

Öncelikle değişkeninizin ismini belirlemek zorundasınız ve değişkenlerinize isim vermek konusunda da bir yere kadar serbestsiniz. Değişken isimlendirme kuralları da programlama dilerinden programlama dillerine göre değişmektedir. Bu ünitede herhangi bir programlama diline bağlı olmadan problemlerimizin çözümüne dair işlem basamaklarını oluşturmaya çalıştığımızdan dolayı bu kurallara şimdilik takılmamıza gerek yoktur.

Sayılarımızı okutacağımız değişkenleri A ve B olarak belirleyelim. O halde A sayısını kullanıcıdan aldırmak için (ve birazda bilgisayar diline yakın olmasını istediğimiz için) "A sayısını oku" gibi bir ifade yeterlidir.

Algoritmamızın bilgisayar diline biraz daha yaklaştırılmış hali aşağıdaki gibidir:

1. Başla
2. A ve B sayılarını oku
3. A ve B sayılarının toplamlarını hesapla
4. Toplamlarını ekrana yaz
5. Dur

Hatta ilgili operatörlerden ("=" aktarma ve "+" toplam operatöründen) yararlanarak algoritmamızı aşağıdaki şekilde yazarsak, programlama dillerine aktarırken işimizi çok kolaylaştırmış oluruz (Çünkü algoritmamızı tam olarak bilgisayarların anlayabileceği bir dille yazmış oluruz)

1. Başla
2. A sayısını oku
3. B sayısını oku
4. C = A + B
5. C sayısını yaz (ekrana yaz)
6. Dur

Burada, algoritmamızın 4. adımında iki operatörden ve bir de üçüncü bir değişkenden yararlandık.

Kullandığımız ilk operatör matematikten de bildiğimiz "+" toplama operatörüdür. Kendisine bağlı operandları (burada A ve B sayısı) toplama işlemine tabi tutar.

Kullandığımız ikinci operatör ise matematikteki eşittir operatörüne benzese de biz programcılar onu "aktarma operatörü" olarak tanımaktayız. Aktarma eşittirden farklıdır. Çünkü matematikte;

x = (x+1)

yazımı imkansızken, programcılıkta en çok kullanacağımız ifadelerden bir tanesidir. Matematikte sağdaki ifade soldaki ifadeye eşittir anlamına gelirken, programcılıkta ise sağdaki ifadeyi soldakine aktarmak anlamına gelmektedir. Önce sağdaki (parantez içindeki) ifade hesaplanır, oluşan değer soldaki değişkene aktarılır. Dolayısıyla bu adımdan sonra "x" değişkenin değeri önceki değerinin bir fazlası olur. (Bu adıma gelmeden önce "x" değişkenin değerinin 2 olduğu varsayılırsa bu adım geçtikten sonra "x" değişkeninin değeri 3 olacaktır. Çünkü program bu adıma geldiğinde "x=x+1" ifadesinin sağ tarafı yani "x+1" hesaplanır. x=2 olduğuna göre x+1 = 3 olur. Hesaplanan bu 3 değeri soldaki ifadeye yani "x" değişkenine aktarılır. Dolayısıyla "x" bu adımdan sonra 3 olarak yoluna devam edecektir).

Yukarıdaki algoritmamızın 4. adımında ki "C = A+B" ifadesinde A ve B sayıları toplanarak oluşan değer yeni bir değişkenin (C değişkenin) içerisine konmuştur. Buradaki amaç, bir değişken vasıtasıyla A ve B sayılarının toplamını 5. adıma aktarabilmektir. 5. adımda da "C" değişkenin değeri dolayısıyla A ve B sayılarının toplamı ekrana yazılmaktadır.

1.7. 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.

ÖZET

Bu ünitede program yazmaya başlamadan önce problemlerimizi tanımlayıp, problemlerimize dair algoritmalarımızı kurmamız gerektiğinden bahsedildi.

Algoritmalar problemlerimizin çözüm aşamalarıydı. Bu çözüm aşamalarını daha da belirgin hale getirip, bilgisayarların anlayabileceği bir dile yaklaştırmak, algoritma adımlarını oluşturmak anlamına geliyordu.

Ayrıca bu ünitede algoritmaların taşıması gereken özelliklerden, bilgisayar programının ve bir algoritmanın çalıştırılmasının ne demek olduğunda da bahsedildi.

GÖZDEN GEÇİR

1. Algoritma nedir? Algoritmanın gerekliliğini ve avantajlarını açıklayınız.
2. Algoritma hazırlanırken dikkat edilmesi gereken hususları, gerekçeleri ile birlikte tartışınız.
3. Değişken nedir? Programlarımızda neden değişkenlere ihtiyaç duymaktayız?

KAYNAKLAR

by. (2007). Programlamaya Giriş ve Algoritmalar Ders Notları.