Programlamaya Giriş

Algoritma ve Programlama 1

Hüseyin Ahmetoğlu

Algoritmalar

- Herhangi bir sorunu çözmek için bir plana ihtiyaç vardır.
- Bu plan, söz konusu sorunu çözmek için bir prosedürdür.
- Prosedür, bir sonuç elde etmek için kesin muhakeme ve mantığa dayanmalıdır.

Algoritma nedir?

- Program = Algoritmalar + Veriler
- Bir algoritma, bilgisayar programı planının bir parçasıdır.
- Aslında, bir algoritma, "bir problemi sınırlı sayıda adımda çözmek için etkili bir prosedürdür".
- Etkilidir, yani bir yanıt bulunur ve sonlu sayıda adımı vardır.
- İyi tasarlanmış bir algoritma her zaman bir cevap verecektir; istenen cevap olmayabilir ama bir cevap olacaktır.
- Cevap, cevap olmaması olabilir. İyi tasarlanmış bir algoritmanın da sonlandırılması garanti edilir.

Algoritmaları Belirtmenin Farklı Yolları

- Adım Formu / Step-form
- Sözde Kod / Pseudo-code
- Akış Şeması / Flowchart
- Nassi–Shneiderman diagramı / N-S Diyagrams

Adım Formunun Temel Özellikleri

- Bir demlik çay yapmak için bir algoritma örneği.
- 1. Su ısıtıcısında su yoksa, su ısıtıcısını doldurun.
- 2. Su ısıtıcısını güç noktasına takın ve açın.
- 3. Çaydanlık boş değilse, demliği boşaltın.
- 4. Çay yapraklarını demliğe yerleştirin.
- 5. Su ısıtıcısının içindeki su kaynamıyorsa, 5. adıma gidin.
- 6. Su ısıtıcısını kapatın.
- 7. Su ısıtıcısından çaydanlığa su dökün.

Bir algoritmanın temel yapıları

- Sıralı Akış / Süreç
- Karar verme / Seçme
- Tekrarlama / Yineleme / Döngü

Sıralı Akış / Süreç

- Sıra, algoritmadaki her adımın veya işlemin belirtilen sırayla yürütülmesi gerekir.
- Çay demleme örneğindeki adımların, her biri uygun yerde olmalıdır, aksi takdirde algoritma başarıyla sonuçlanmaz.

Karar Yapıları—`if ... then, if ... then ... else...`

- Algoritmalarda bir kararın sonucu ya doğru ya da yanlıştır. Arada karar yok.
- Kararın sonucu, yalnızca doğru veya yanlış bir değerle sonuçlanabilecek bazı koşullara dayanmaktadır.
  - if bugün ayın 15'mi? then Maaş al!

```
if proposition
        then process1
else process2
```

Tekrarlama Yapıları – `repeat and while`

- Yineleme döngüsü, bir işlemi veya bir adımı yinelemek veya tekrarlamak için kullanılır.
- Bazı koşullar gerçekleşene kadar işlem dizisi tekrarlanır.

```
Repeat
        Process1
        Process2
        ..............
        ...........
        ProcessN
        Until proposition
```

```
while proposition
begin
Process 1
Process 2
...........
...........
Process N
end
```

```
if .. then goto .. is:
Process1
Process2
..........
..........
ProcessN
if proposition then goto Process1
```

Değişken nedir?

- Neredeyse her algoritma veri içerir ve genellikle veri, değişken adı verilen şeyin içinde bulunur.
- Değişken, programın yürütülmesi sırasında değişebilen bir değer için bir saklama alanını temsil eder.
- Örneğin çay yapma algoritmasında su ısıtıcısındaki su seviyesi değişkendir, suyun sıcaklığı değişkendir ve çay yapraklarının miktarı da değişkendir.

Değişken türleri

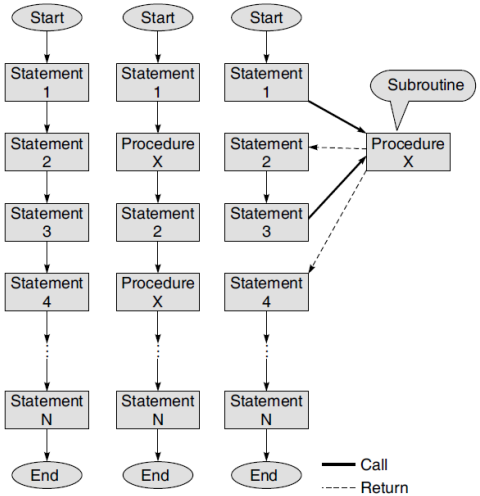
- Algoritmalarda kullanılan veriler farklı türlerde olabilir.
- Bir algoritmanın kullanabileceği en basit veri türleri
  - sayısal veriler, ör. 12, 11.45, 901 vb.
  - "A", "Z" veya "Bu" gibi alfabetik veya karakter verileri’
  - mantıksal veriler, yani doğru / yanlış değerlere sahip önermeler

Değişken adlandırma

- Algoritma veya programın okunabilirliğini geliştirmek için her zaman algoritmalardaki değişkenler için anlamlı isimler seçmeye çalışılmalıdır.
- Bu, özellikle büyük ve karmaşık programlarda önemlidir.
- Değişkenlerin nasıl adlandırılması gerektiğine dair kesin kurallar yoktur, ancak birçok kural vardır. Değişkenleri adlandırmanın geleneksel bir yolunu kullanmak kodların diğer yazılımcılara hitap edebilmesi için önemlidir.

Altyordamlar / Prosedürler / Fonksiyonlar / Metotlar

- Basit bir program, sıralı bir sırada uygulanan ifadelerin bir kombinasyonudur. Bir ifade bloğu, bir ifadeler grubudur.
- Daha büyük programın uygulama sırasındaki farklı noktalarda birkaç kez çalıştırılan bir prosedür olarak da bilinen belirli bir ifade bloğu olabilir.



Altyordamlar / Prosedürler / Fonksiyonlar / Metotlar

- Prosedür, fonksiyon veya metot olarak da bilinen bir alt yordam, belirli bir görevi gerçekleştirmek için daha büyük bir programın içinden çağrılan talimatın bir bölümüdür.
- Aynı zamanda altyordam, daha büyük programın kalan ifadelerinden nispeten bağımsızdır.
- Altyordam, daha büyük bir programda bir adım olarak kullanılan bir programla aynı şekilde davranır.
- Bir alt yordam genellikle, diğer alt yordamlar da dahil olmak üzere, programın tek bir yürütmesi sırasında birkaç kez ve / veya birkaç yerden çağrılabilir (called) ve daha sonra, Alt programın görevi tamamlandığında "çağrı" diğer alt yordamlar da dahil olmak üzere, bir sonraki talimata geri döner (return).
- Böylelikle, bu tür alt yordamlar, çağıran programdan parametreler geçerek veya geçmeden bir CALL ifadesiyle çağrılır.
- Alt rutin, kendisine verilmişse parametreler üzerinde çalışır, aksi takdirde sonuçları hesaplar ve sonucu kendisi verir ve çağıran programa geri döner veya sonuçları geri döndürmeden önce çağıran programa iletir.

Fonksiyon yazmanın avantajları

- Alt yordam, bir program içindeki ifade bloklarının tekrarlanmasını azaltır,
- alt yordamı birden çok programda oluşturan ifadeler bloğunun yeniden kullanılmasını sağlar,
- karmaşık bir görevi daha basit adımlara dönüştürür,
- büyük bir programlama görevini çeşitli programcılar veya bir projenin çeşitli aşamaları arasında böler ve uygulama ayrıntılarını kullanıcılardan gizler.

Adım Formu için kurallar

- 1. Her algoritma mantıksal olarak START ve STOP ifadeleri ile çevrelenecektir.
- 2. Kullanıcıdan veri kabul etmek için INPUT veya READ deyimleri kullanılacaktır.
- 3. Herhangi bir kullanıcı mesajını veya bir değişkendeki içeriği görüntülemek için, PRINT deyimi kullanılacaktır. Mesajın tırnak içine alınacağını unutmayın.
- 4. Bir algoritmada birkaç adım vardır. Her adım bir eylemle sonuçlanır. Adımlar, düzenlendikleri veya yönlendirildikleri sırayla uygulanmalıdır.

Aritmetik Operatörler

- (←)…. Atama (‘←’ nın sol tarafı her zaman tek bir değişken olmalıdır)
  - Örnek: x ← 6 ifadesi, x değişkenine 6 değerinin atandığı anlamına gelir. Bellek depolama açısından, değişken x'e tahsis edilen bellekteki bir konumda 6 değerinin saklandığı anlamına gelir.
- (+)… .. toplama
  - Örnek: z ← x + y ifadesi, x değişkeninde bulunan değer, y değişkeninde bulunan değere eklenir ve elde edilen sonuç z değişkenine atanır.
- (-)… .. Çıkarma
  - Örnek: z ← x - y ifadesi, y değişkeninde bulunan değerin x değişkeninde bulunan değerden çıkarıldığı ve elde edilen sonuç değerinin z değişkenine atandığı anlamına gelir.

Aritmetik Operatörler

- (*)….. Çarpma işlemi
  - x ← 5
  - y ← 6
  - z ← x * y
  - X ve y arasındaki çarpmanın sonucu 30'dur. Bu nedenle bu değer z'ye atanır.
- (/)….. Bölünme
  - x ← 10
  - y ← 6
  - z ← x / y
  - X ve y arasındaki bölümün oranı 1 ve kalanı 4'tür. Böyle bir operatör kullanıldığında bölüm sonuç olarak alınır, kalanı reddedilir. Yani burada x / y ifadesinden elde edilen sonuç 1'dir ve bu z'ye atanmıştır

İlişkisel Operatörler

- (>) … Büyük mü?
- (<) … Küçük mü?
- (<=) … Küçük veya eşit mi?
- (>=) … Büyük veya eşit mi?
- (=) … Eşit mi?
- (!=) … Eşit değil mi?

  - x> y ifadesi, x'in içerdiği değer y'deki değerden daha büyükse, ifadenin sonucu doğrudur ve bu 1 olarak alınacaktır. Aksi takdirde, sonuç yanlışsa, 0 olarak alınacaktır.

Mantıksal Operatörler

- (AND)
- Ve ile bağlanmış ifadelerin sonucunun 1 olabilmesi için tüm ifadelerin doğru sonuç döndürmesi gerekir.
  - x ← 2
  - y ← 1
  - x = 2 AND y = 0
- Yukarıdaki ifadede 'x = 2' önermesi doğrudur çünkü x'teki değer 2'dir. Benzer şekilde, y = 1'i tuttuğundan 'y = 0' önermesi yanlıştır ve bu nedenle bu öneri yanlış veya 0'dır. Yanlış veya 0 olan sonuç ‘true’ AND false' olarak temsil edilebilir.

Mantıksal Operatörler

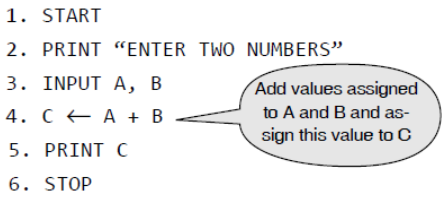
- (OR)
- VEYA ile bağlanmış ifadelerin sonucunun 1 olabilmesi için ifadelerin herhangi birinin doğru sonuçla dönmesi yeterlidir.
  - x ← 2
  - y ← 1
  - x = 2 OR y = 0
- Yukarıdaki ifadede 'x = 2' önermesi doğrudur çünkü x'teki değer 2'dir. Benzer şekilde, y = 1'i tuttuğundan 'y = 0' önermesi yanlıştır ve bu nedenle bu öneri yanlış veya 0'dır. ‘True’ OR ‘False’ ifadesinin sonucu True yani 1’dir.

Mantıksal Operatörler

- (NOT)
- Bir önermenin sonucu "doğru" ise, terslendiği veya DEĞİL'i alındığı zaman "yanlış" olur.
  - x ← 2
  - NOT x = 2
- 'x = 2' önermesi, x değeri 2'yi içerdiğinden 'doğru'dur. Ancak bu ifade Not operatörü tarafından terslenmiştir yani DEĞİL'i alınmıştır ve 'doğru' olan önerme 'yanlış' sonucu döndürmüştür.

Örnekler

- İki sayıyı toplayıp sonucu ekrana yazdıran program



Örnekler

- Klavyeden girilen iki sayı için bölme işlemini yapan ve kalanı ekrana yazdıran program.

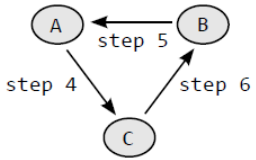
```
1. START
2. PRINT "ENTER DIVIDEND"
3. INPUT N
4. PRINT "ENTER DIVISOR"
5. INPUT D
6. Q ← N/D (Integer division)
7. R ← N - Q * D
8. PRINT R
9. STOP
```

Only integer value is obtained and remainder ignored

Örnekler

- Klavyeden girilen iki değişkenin değerlerini değiş tokuş yapan program

```
1. START
2. PRINT “ENTER THE VALUE OF A & B”
3. INPUT A, B
4. C ← A
5. A ← B
6. B ← C
7. PRINT A, B
8. END
```



Örnekler

- Kullanıcı tarafından girilen iki değişkenden büyük olanını ekrana yazdıran program

```
1. START
2. PRINT “ENTER TWO NUMBERS”
3. INPUT A, B
4. IF A > B THEN
              PRINT “A IS GREATER THAN B”
5. IF B > A THEN
              PRINT “B IS GREATER THAN A”
6. IF A = B THEN
              PRINT “BOTH ARE EQUAL”
7. STOP
```

Örnekler

- Kullanıcı tarafından girilen bir sayının tek mi, çift mi olduğunu bulan program

```
1. START
2. PRINT “ENTER THE NUMBER”
3. INPUT N
4. Q ← N/2 (Integer division)
5. R ← N — Q * 2
6. IF R = 0 THEN
                PRINT “N IS EVEN”
7. IF R != 0 THEN
                PRINT “N IS ODD”
8. STOP
```

Örnekler

- Kullanıcı tarafından girilen üç sayıdan en büyüğünü ekrana yazdıran program

```
1. START
2. PRINT “ENTER THREE NUMBERS”
3. INPUT A, B, C
4. IF A >= B AND B >= C
            THEN PRINT A
5. IF B >= C AND C >= A
            THEN PRINT B
            ELSE
                    PRINT C
6. STOP
```

```
1. START
2. PRINT “ENTER THREE NUMBERS”
3. INPUT A, B, C
4. MAX ← A
5. IF B > MAX THEN MAX ← B
6. IF C > MAX THEN MAX ← C
7. PRINT MAX
8. STOP
```

```
1. START
2. PRINT “ENTER THREE NUMBERS”
3. INPUT A, B, C
4. IF A > B THEN
    IF A > C THEN
     PRINT A
    ELSE
     PRINT C
    ELSE IF B > C THEN
     PRINT B
    ELSE
     PRINT C
5. STOP
```

Ödevler

- 1. Kullanıcı tarafından 3 kenar uzunluğu girilen bir üçgenin çizilebilir olup olmadığını bulan, eğer çizilebilir ise bu üçgenin türünü veren programın algoritmasını yazınız. (Eşkenar-ikizkenar-çeşitkenar)
- 2. Kullanıcı tarafından verilen vize ve final notuna göre başarı notunu hesaplayıp bu notun HARF NOTU karşılığını bulan ve ekrana yazdıran programın algoritma adımlarını yazınız.
- 3. İlk değeri 1 olan değişkenin sürekli değerini 1 artıran ve bu değerleri ekrana yazan, değişkenin değeri 100 olunca sonlanan programın algoritma adımlarını yazınız.
- 4. Kullanıcı tarafında girilen N adet sayıyı toplayan ve sonucu ekrana yazan programın algoritma adımlarını yazınız.
- 5. Kullanıcı tarafından girilen bir sayının basamaklarındaki rakamları toplayıp sonucu ekrana yazdıran programın algoritma adımlarını yazınız.

KAYNAKLAR

- Goel, A., & Mittal, A. (2016). Computer Fundamentals and Programming in C (RMK). Pearson Education India. Retrieved from https://www.oreilly.com/library/view/computer-fundamentals-and/9789332579200