Pseudocode

Bir Algoritmayı Temsil Etme: Pseudocode

Pseudocode Nedir?

Bir algoritmayı temsil etmenin iki ana yolu vardır: **Pseudocode** ve **flowchart**.

Pseudocode, bir bilgisayar programı veya algoritmasının resmi olmayan üst düzey bir açıklamasıdır. Çalıştırılmadan önce bir programlama diline çevrilmesi gereken sembolik kodla yazılmıştır. Pseudocode kullanmak, "bir programlama dilinde yazmaya" benzer ve şöyle görünebilir:

```
OUTPUT 'What is your name?'
INPUT user inputs his/her name
STORE the user's input in the name variable
OUTPUT 'Hello' + name
OUTPUT 'How old are you?'
INPUT user inputs his/her age
STORE the user's input in the age variable
IF age >= 70 THEN
    OUTPUT 'You are aged to perfection!'
ELSE
    OUTPUT 'You are a spring chicken'
```

INPUT bir soru sorar. OUTPUT ekrana bir mesaj yazdırır.

Pseudocode, program oluşturmayı kolaylaştırır. Programlar karmaşık ve uzun olabilir; hazırlık anahtardır. Yıllar boyunca, bir dilde bir satır kod yazmadan önce programları haritalamak için akış şemaları kullanıldı. Ancak bunları değiştirmek zordu ve programlama dillerinin gelişmesiyle bir programın tüm bölümlerini bir akış şemasıyla görüntülemek zorlaştı. Bir programın tam akışını anlamadan bir hata bulmak zordur. Pseudocode daha çekici hale geldiği yer burasıdır.

Pseudocode kullanmak için, programınızın ne söylemesini istediğinizi İngilizce olarak yazmanız yeterlidir. Pseudocode, özel komutlar olmadığı ve standartlaştırılmadığı için ifadelerinizi herhangi bir dile çevirmenize olanak tanır. Programları kodlamadan önce yazmak, daha iyi organize olmanızı ve programlarınızda gerekli bölümleri nerede atlamış olabileceğinizi görmenizi sağlayabilir. Tek yapmanız gereken, kısa ifadelerle kendi kelimelerinizle yazmak.

Bazı örneklere bakalım.

```
Begin
INPUT hours
INPUT rate
pay = hours * rate
OUTPUT pay
End
```

Ve bu, biraz daha karmaşık örnek, fazla mesai ile ödemeyi hesaplayabilir:

```
Begin
INPUT hours, rate
IF hours < 40
THEN
    pay = hours * rate
ELSE
    pay = 40 * rate + (hours - 40) * rate *1.5
OUTPUT pay
End
```

Çoğu program, programlama dilleri kullanılarak geliştirilir. Bu dillerin, programın düzgün çalışması için kullanılması gereken belirli bir sözdizimi vardır. Pseudocode bir programlama dili değildir, belirli bir sözdizimi kullanmak zorunda olmayan bir dizi talimatı tanımlamanın basit bir yoludur.

Bir Pseudocode yazmak, bir programlama dilinde yazmaya benzer. Algoritmanın her adımı sırayla kendi satırına yazılır. Genellikle talimatlar büyük harfle, değişkenler küçük harfle ve mesajlar cümle halinde yazılır.

Neden Pseudocode kullanılmalıdır?

Prototip, kavram testi ve öğrenme amacıyla oluşturulan bir ürünün erken örneği, modeli veya sürümüdür. Çözümlerimizi tam olarak uygulamadan öğrenmemize yardımcı olurlar. Uygulamalarımız için kullanıcı arayüzleri geliştirirken, son arayüzden önce birkaç prototipimiz var. Bunların bazı örnekleri wire-frameler, grafik tasarımlar ve maketlerdir. Aynısı teknik kod yazmak için de geçerlidir. Karmaşık amaçlar için doğrudan kod yazmak zaman kaybına neden olabilir. Bunun nedenleri, uygun olmayan algoritmalardan belirsiz program akışına kadar uzanır. Bunu önlemek için Pseudocode kullanabiliriz.

Sözde kodun avantajları:

- Sözde kod, her türden programcı tarafından anlaşılır.
- Programcının sadece kod geliştirmenin algoritma kısmına konsantre olmasını sağlar.
- Yürütülebilir bir programda derlenemez.

Pseudocode Nasıl Yazılır – 1

Statements

Statement(İfade), bilgisayarı belirli bir eylemi gerçekleştirmeye yönlendiren bir talimat olarak tanımlanır. Pseudocode yazarken, tekil talimatlara ifadeler olarak atıfta bulunacağız. Pseudocode yazarken, ifadelerin yürütme sırasının yukarıdan aşağıya doğru olduğunu varsayıyoruz. Bu, kontrol yapıları (control structures), işlevler (functions) ve istisna işleme (exception handling) kullanılırken değişir.

Matematiksel işlemler, çözüm geliştirmenin ayrılmaz bir parçasıdır. Sakladığımız değerleri manipüle etmemize izin veriyorlar. İşte yaygın matematiksel semboller:

```
Assignment: ← or :=
    Example: c ← 3, c := 2
Comparison: =, ≠, <, >, ≤, ≥
Arithmetic: +, −, ×, /, mod
Logical: and, or
```

Anahtar Sözcükler (**Keywords**) Anahtar sözcük, özel bir anlamı olduğu için bir program tarafından ayrılmış bir sözcüktür. Anahtar kelimeler komutlar veya parametreler olabilir. Her programlama dilinin kendi anahtar kelimeleri (reserved words/ayrılmış kelimeler) vardır. Anahtar kelimeler değişken adı olarak

kullanılamaz. Pseudocode'da ortak giriş-çıkış (input-output) ve işleme (processing) işlemlerini belirtmek için kullanılırlar. Tamamen büyük harfle yazılırlar.

START, BEGIN: Pseudocode'un başlangıcıdır.
INPUT: Kullanıcıdan giriş cihazı aracılığıyla alınan verilerdir.
READ, GET: Veri dosyasından veri okunurken kullanılır.
PRINT, DISPLAY, SHOW, OUTPUT: Çıktıyı ekrana gösterir.
COMPUTE, CALCULATE: İfadenin sonucunu hesaplar.
INCREMENT, BUMP: Bir değişkenin değerini artırır.
DECREMENT: Bir değişkenin değerini düşürür.
END: Pseudocode'un sonudur.

Conditionals (Şartlılar)

Algoritma geliştirme sırasında, ifadelerin Doğru veya Yanlış olarak değerlendirilmesine bağlı olarak ifadeleri değerlendiren ve yönergeleri yürüten ifadelere ihtiyacımız var. Pseudocode'da kullanılan bazı genel koşullar şunlardır:

IF – ELSE IF – ELSE

Bu, belirli bir koşul karşılandığında yürütülecek ifadeleri sağlamak için kullanılan bir koşuldur. Aynı zamanda birden çok koşul ve farklı değişkenler için de geçerlidir. İşte tek koşullu bir if ifadesi;

```
IF you are happy
   THEN smile
ENDIF
```

İşte else bölümü olan bir if ifadesi. Else, “if” koşulu sağlanmadığında bazı ifadelerin yürütülmesine izin verir.

```
IF you are happy
   THEN smile
ELSE
    frown
ENDIF
```

Karşılanırsa farklı ifadeleri yürütmek için ek koşullar ekleyebiliriz.

```
IF you are happy
   THEN smile
ELSE IF you are sad
    THEN frown
ELSE
    keep face plain
ENDIF
```

CASE

Tek bir değişkeni birkaç koşulla karşılaştırmak istiyorsak CASE kullanılır.

```
INPUT color
CASE color of
    red: PRINT "red"
    green: PRINT "green"
    blue: PRINT "blue"
OTHERS
    PRINT "Please enter a value color"
ENDCASE
```

OTHERS ifadesi isteğe bağlıdır. Koşullar normalde sayılar veya karakterlerdir.

Iteration

Yinelemek (Iterate), bir dizi sonuç üretmek için bir dizi talimatı tekrarlamaktır. Belirli bir hedefe ulaşabilmek için yineleniriz.

FOR

FOR döngüsü bir grup elemanı alır ve her eleman için döngü içinde kodu çalıştırır.

```
FOR every month in a year
    Compute number of days
ENDFOR
```

WHILE

FOR döngüsüne benzer şekilde, while döngüsü, önceden tanımlanmış bir koşul doğru kaldığı sürece bir kod bloğunu tekrarlamanın bir yoludur. FOR döngüsünden farklı olarak while döngüsü, koşulun ne kadar süreyle doğru kalacağına göre değerlendirilir.

Döngümüzün sonsuz olarak çalıştığı bir senaryodan kaçınmak için, her yineleme içindeki değeri değiştirmek için bir işlem ekleriz. Bu, bir artış, azalma vb. olabilir.

```
PRECONDITION: variable X is equal to 1
WHILE Population < Limit
    Compute Population as Population + Births –
Deaths
ENDWHILE
```

Functions

Gelişmiş görevleri çözerken, kavramları farklı konumlarda bir ifadeler bloğuna bölmek gerekir. Bu, özellikle söz konusu ifadeler belirli bir amaca hizmet ettiğinde geçerlidir. Bu kodu yeniden kullanmak için işlevler oluşturuyoruz. Daha sonra, çalıştırılmalarına ihtiyaç duyduğumuz her zaman bu işlevleri çağırabiliriz.

```
Function clear monitor
  Pass In: nothing
  Direct the operating system to clear the monitor
  Pass Out: nothing
Endfunction
```

Bir işlev çağrısını pseudocode'da taklit etmek için Call anahtar sözcüğünü kullanabiliriz.

Program Sarma (Program Wrapping)

Pseudocode'da birkaç fonksiyon yazdıktan sonra, her şeyi tek bir programın içine almamız gerektiğini görüyoruz. Bu, okunabilirliği geliştirmek ve akışın yürütülmesinin anlaşılmasını kolaylaştırmak içindir.

Bunu yapmak için kodumuzu bir programq sarıyoruz. Bir program, yürütüldüğünde belirli bir görevi yerine getiren bir dizi talimat olarak tanımlanabilir.

```
PROGRAM makeacupoftea


END
```

İstisna işleme (Exception Handling)

İstisna, program yürütme sırasında meydana gelen ve talimatların normal akışını bozan bir olaydır. Bunlar istenmeyen olaylardır.

Bu tür olayları gözlemlememiz ve bunlara yanıt olarak kod blokları yürütmemiz gerekiyor. Buna istisna işleme denir.

```
BEGIN
    statements
EXCEPTION
    WHEN exception type
        statements to handle exception
    WHEN another exception type
        statements to handle exception
END
```

Sonuç

Pseudocode için teknik kurallar yoktur. İnsan tarafından okunabilir olması ve yine de anlam ve akış iletmesi amaçlanmıştır.

Daha çok dile özgü farklı kılavuz ve öğreticileri olan pseudocode vardır, bunların örnekleri Fortran stili pseudocode, Pascal stili pseudocode, C stili pseudocode ve Structured Basic stil pseudocode dur.

Pseudocode Nasıl Yazılır - 2

Artık pseudocode un nasıl göründüğünü bildiğinize göre, nasıl yazılacağının temel yönlerini öğrenmenin zamanı geldi.

Nasıl yazılacağını ve etkili bir şekilde nasıl kullanılacağını tam olarak bilmeniz için pseudocode yazma sürecini adım adım anlatacağız.

1. Kullanımları Anlayın

Pek çok kullanımını gerçekten anlamadıysanız, pseudocode u kullanmak zordur.

Yeni başlayanlar için, pseudocode, yeni bir bilgisayar programı oluşturma görevini daha basit ve anlaşılır hale getirir.

Kodu İngilizce olarak yazmak, projenin programlama aşamalarında izlenecek sözlü bir taslak oluşturmanıza olanak tanır.

Pseudocode, programlama sırasında ihtiyacınız olan her şeyin dahil edilmesini sağlamak için gereken araçları sağlar. Hataları hataya dönüşmeden yakalamanızı sağlar.

Grup projeleri için pseudocode kullanmak da oldukça faydalıdır. Tüm programcıların aynı sayfada olması için programı basit bir şekilde parçalar.

2. Pseudocode Özneldir

Pseudocode ile ilgili en zor şey, öznel olmasıdır.

Pseudocode yazmanın standart bir yolu yoktur. Amaç basitçe aklınızdaki her şeyi doğru bir şekilde özetlemektir.

Bununla birlikte, başkalarıyla çalışıyorsanız kullanmanız gereken belirli yapılar ve standart prosedürler vardır. Ekipteki diğer herkesin aynı sayfada olmasını sağlamak için bu kuralları izleyin.

Belki de en önemli kural, önce netliği yerleştirmektir. Pseudocode unuzu mümkün olduğunca açık ve özlü yapın, böylece ne demek istediğiniz konusunda hiçbir şüpheniz olmasın.

3. Algoritmalar ve Temel Yapılar

Pseudocode yazmaya gelince anlamanız gereken en önemli şeylerden ikisi algoritmalar ve temel algoritma yapılarıdır.

Algoritmaları Anlayın – Bir algoritma, belirli bir hedefe ulaşmak için atmanız gereken adımlardır.

Algoritma Akışını Bilin – En temel algoritma yapısı veya akışı “sıralama”, “seçim” ve “yineleme”dir. Bunlar, kodu yazmanın doğru yolunu gösterir.

Parçaları Birleştirin – Aktarmak istediğiniz bilgileri alın ve basit bir anahat oluşturmak için algoritma akışını kullanın.

4. Standart Prosedür

Yukarıda belirtildiği gibi, Pseudocode yazmak için standart bir prosedür yoktur. Ancak bu, uymanız gereken belirli kurallar olmadığı anlamına gelmez.

İşbirliği yaptığınız herkesin Pseudocode unuzu anladığından emin olmak için bu temel kuralları izleyin.

Satır Başına Bir İfade – Her bir ifadeyi veya eylemi kendi satırında ifade edin.

Talimatları Büyük Harf Yap – Önemlerini vurgulamak için talimatları büyük harfle yaz (örneğin, “READ”).

Anlama Odaklan – Programın ne yapacağını yazın. Nasıl programlanacağını yazmayın.

Standart Programlama Yapıları – Takip etmesi kolay yapılar oluşturmak için yukarıda tartışılan algoritma akışını izleyin.

Blokları Kullan – Sözde kodu ayrı adımlara ayırmak için benzer eylemleri bloklar halinde gruplayın.

5. Önemli İpuçları

Bir kez daha, Pseudocode un katı ve hızlı kuralları olmasa da, Pseudocode un herkes için anlaşılmasını kolaylaştırmak için kesinlikle yapmanız gereken bazı şeyler vardır.

Basit Tutun – Basitlik ve netlik anahtardır. Eylemlerin ne olacağını yazın, nasıl programlanacağını değil.

Her Şeyi Açıklayın – Bilgiyi açıklamadan dahil etmeyin. Adımlarınızı ve gerekirse gerekçelerinizi açıklamak için yorumlar ekleyin.

Alıştırma Mükemmelleştirir – Tıpkı yeni bir programlama dili öğrenmek gibi, sözde kod yazmayı öğrenmek de zaman alır. Şimdi yazma ve gözden geçirme alıştırması yapın.

Pseudocode gözden geçirin – Pseudocode yazmanın en büyük nedeni, programlamadan önce herhangi bir hatayı yakalamaktır. Bu nedenle, tomurcuktaki hataları bulmak için bitmiş ürünü iyice gözden geçirin.

Programlama Dili'ne Çevir - Pseudocode bilgisayar dilinizi dikkate alarak yazın. Bitmiş ürünü Pseudocode ile karşılaştırın.

Pseudocode: Alıştırma

Alıştırma 1: İki sayıyı okuyan ve bunları çarparak çarpımlarını yazdıran bir Pseudocode yazın.

```
Begin
INPUT number1 , number2
SET answer to num1 * num2
OUTPUT answer
End
```

Alıştırma 2: Bir kullanıcıya girdiği sayının 5 veya 6 olmadığını söyleyen bir Pseudocode yazın.

```
Begin
INPUT number
IF (number = 5)
    OUTPUT "your number is 5"
ELSE IF (number = 6)
    OUTPUT "your number is 6"
ELSE
    OUTPUT "your number is not 5 or 6"
End
```

Alıştırma 3: 0 ile 100 arasında (hem 0 hem de 100 dahil) 5'in tüm katlarını yazdıracak bir Pseudocode yazın.

(YANLIŞ)

```
Begin
SET x to 0
WHILE (x <= 20)
    OUTPUT x * 5
    x = x + 1
End
```

Alıştırma 4: Kullanıcı tanımlı bir durma noktasına kadar tüm çift sayıları sayacak bir Pseudocode yazın.

Örneğin, 0'dan başlayan ilk 5 çift sayıyı görmek istediğimizi varsayalım.

Çift sayıların 0, 2, 4 vb. olduğunu biliyoruz.

İlk 5 çift sayı 0, 2, 4, 6, 8'dir.

İlk 8 çift sayı 0, 2, 4, 6, 8 ,10 ,12, 14'tür.

```
Begin
INPUT stopping_point
SET x to 0
SET even to 0

WHILE (x < stopping_point)
    OUTPUT even
    x = x + 1
    SET even to even + 2

End
```

Alıştırma 5: Aşağıdakileri gerçekleştirecek bir Pseudocode yazın.

5 ayrı sayı okuyun.

Beş sayının ortalamasını hesaplayın.

Girilen beş sayıdan en küçüğünü (en küçük) ve en büyüğünü (en büyük) bulun.

2. ve 3. adımlarda bulunan sonuçları, ne olduklarını açıklayan bir mesaj yazın.

```
Begin
OUTPUT "please enter 5 seperate numbers"
INPUT n1, n2, n3, n4, n5
OUTPUT "The average is:"
SET avg to (n1 + n2 + n3 + n4 + n5) / 5
OUTPUT avg

IF (n1 < n2)
    SET max to n2
ELSE
    SET max to n1
IF (n3 > max)
    SET max to n3
IF (n4 > max)
    SET max to n4
IF (n5 > max)
    SET max to n5
OUTPUT "The max is:"
OUTPUT max

IF (n1 > n2)
    SET min to n2
ELSE
    SET min to n1
```

```
IF (n3 < min)
    SET min to n3
IF (n4 < min)
    SET min to n4
IF (n5 < min)
    SET min to n5
OUTPUT "The min is:"
OUTPUT min

End
```