Pseudocode

Representing an Algorithm: Pseudocode

* **What is Pseudocode?**

There are two main ways to represent an algorithm: Pseudocode and flowchart. Pseudocode is an informal high-level description of a computer program or algorithm. It is written in symbolic code which must be translated into a programming language before it can be executed. Using Pseudocode is similar to “writing in a programming language” and might look something like this:

Bir algoritmayı temsil etmenin iki ana yolu vardır: Sözde kod ve akış şeması. Sahte kod, bir bilgisayar programının veya algoritmanın resmi olmayan yüksek düzeyli bir tanımıdır. Yürütülmeden önce bir programlama diline çevrilmesi gereken sembolik kodda yazılmıştır. Pseudocode kullanımı “bir programlama dilinde yazmak” ile benzerdir 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*** asks a question. ***OUTPUT*** prints a message on the screen.

Pseudocode makes creating programs easier. Programs can be complex and long; preparation is the key. For years, flowcharts were used to map out programs before writing one line of code in a language. However, they were difficult to modify and with the advancement of programming languages, it was difficult to display all parts of a program with a flowchart. It is challenging to find a mistake without understanding the complete flow of a program. That is where pseudocode becomes more appealing.

Sözde kod, program oluşturmayı kolaylaştırır. Programlar karmaşık ve uzun olabilir; hazırlık anahtardır. Yıllar boyunca, bir dilde bir kod satırı yazmadan önce programları haritalamak için akış şemaları kullanılmıştır. Bununla birlikte, değiştirilmeleri zordu ve programlama dillerinin ilerlemesi ile, bir programın tüm bölümlerini bir akış şeması ile görüntülemek zordu. Bir programın tam akışını anlamadan bir hata bulmak zordur. Bu, sözde kodun daha cazip hale geldiği yerdir.

To use pseudocode, all you do is write what you want your program to say in English. Pseudocode allows you to translate your statements into any language because there are no special commands and it is not standardized. Writing out programs before you code can enable you to better organize and see where you may have left out needed parts in your programs. All you have to do is write it out in your own words in short statements. Let's look at some examples.

Sahte kod kullanmak için tek yapmanız gereken, programınızın İngilizce olarak söylemesini istediğiniz şeyleri yazmaktır. Pseudocode, özel komutlar olmadığı ve standartlaştırılmadığı için ifadelerinizi herhangi bir dile çevirmenize izin verir. Kodlamadan önce programları yazmak, programlarınızda gerekli parçaları nerede bırakabileceğinizi daha iyi organize etmenizi ve görmenizi sağlayabilir. Tek yapmanız gereken kısa ifadelerle kendi kelimelerinizle yazmak. Bazı örneklere bakalım.

Begin

INPUT hours

INPUT rate

pay = hours \* rate

OUTPUT pay

End

And this, slightly more complex example might calculate pay with overtime:

Begin

INPUT hours, rate

IF hours < 40

THEN

pay = hours \* rate

ELSE

pay = 40 \* rate + (hours - 40) \* rate \*1.5

OUTPUT pay

End

Most programs are developed using programming languages. These languages have a specific syntax that must be used so that the program will run properly. Pseudocode is not a programming language, it is a simple way of describing a set of instructions that does not have to use specific syntax.

Çoğu program programlama dilleri kullanılarak geliştirilir. Bu diller, programın düzgün çalışması için kullanılması gereken belirli bir sözdizimine sahiptir. Sözde kod bir programlama dili değildir, belirli bir sözdizimi kullanmak zorunda olmayan bir dizi talimatı tanımlamanın basit bir yoludur.

Writing in a pseudocode is similar to writing in a programming language. Each step of the algorithm is written on a line of its own in sequence. Usually, instructions are written in uppercase, variables in lowercase and messages in sentence case.

Sahte kodda yazma, programlama dilinde yazmaya benzer. Algoritmanın her adımı sırayla kendi satırına yazılır. Genellikle, talimatlar büyük harf, değişkenler küçük harf ve mesajlar cümle halinde yazılır.

* **Why use Pseudocode?**

A prototype is an early sample, model or release of a product created with the intention of concept testing and for learning purposes. They help us to learn without fully implementing our solutions. When developing user interfaces for our applications, we have several prototypes before the final interface. Some examples of these are wire-frames, graphical designs, and mock-ups. The same applies to writing technical code. Directly writing code for complex purposes might result in time wastage. The causes of this range from improper algorithms to ambiguous program flow. To prevent this, we can use Pseudocode.

Advantages of pseudocode:

* Pseudocode is understood by the programmers of all types.
* It enables the programmer to concentrate only on the algorithm part of the code development.
* It cannot be compiled into an executable program.

Prototip, kavram testi ve öğrenme amacıyla oluşturulan bir ürünün erken bir örneğidir, modelidir veya piyasaya sürülmüştür. Çözümlerimizi tam olarak uygulamadan öğrenmemize yardımcı olurlar. Uygulamalarımız için kullanıcı arayüzleri geliştirirken, son arayüzden önce birkaç prototipimiz var. Bunların bazı örnekleri tel çerçeveler, grafik tasarımlar ve maketlerdir. Aynı şey teknik kod yazmak için de geçerlidir. Doğrudan karmaşık amaçlar için kod yazmak zaman kaybına neden olabilir. Bunun nedenleri uygunsuz algoritmalardan belirsiz program akışına kadar değişir. Bunu önlemek için Pseudocode kullanabiliriz.

Sahte kodun avantajları:

Sözde kod, her tür programcı tarafından anlaşılmaktadır.

Programlayıcının sadece kod geliştirmenin algoritma kısmına konsantre olmasını sağlar.

Yürütülebilir bir programda derlenemez.

How to Write Pseudocode - 1

**Statements**

A statement is defined as an instruction that directs the computer to perform a specific action. In writing pseudocode, we will refer to singular instructions as statements. When writing pseudocode, we assume that the order of execution of the statements is from top to bottom. This changes when using control structures, functions and exception handling.

İfade, bilgisayarı belirli bir eylem gerçekleştirmeye yönlendiren bir yönerge olarak tanımlanır. Sözde kod yazarken, tekil talimatlara ifade olarak değineceğiz. Sahte kod yazarken, ifadelerin yürütme sırasının yukarıdan aşağıya olduğunu varsayarız. Bu, kontrol yapıları, fonksiyonlar ve istisna işleme kullanılırken değişir.

* **Mathematical operations** Mathematical operations are integral to solution development. They allow us to manipulate the values we have stored. Here are common mathematical symbols:

Assignment: ← or :=

Example: c ← 3, c := 2

Comparison: =, ≠, <, >, ≤, ≥

Arithmetic: +, −, ×, /, mod

Logical: and, or

* **Keywords** A keyword is a word that is reserved by a program because the word has a special meaning. Keywords can be commands or parameters. Every programming language has its own keywords (reserved words). Keywords cannot be used as variable names. In Pseudocode, they are used to indicate common input-output and processing operations. They are written fully in uppercase.
* **Anahtar sözcükler** 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 vardır (ayrılmış kelimeler). Anahtar kelimeler değişken adı olarak kullanılamaz. Sözde kodda, ortak giriş-çıkış ve işleme işlemlerini belirtmek için kullanılırlar. Tamamen büyük harfle yazılırlar.

START, BEGIN: This is the start of your pseudocode.

INPUT: This is data retrieved from the user through the input device.

READ, GET: This is used when reading data from a data file.

PRINT, DISPLAY, SHOW, OUTPUT: This will show your output to a screen.

COMPUTE, CALCULATE: To calculate the result of the expression.

SET, INIT: To initialize values

INCREMENT, BUMP: To increase the value of a variable

DECREMENT: To reduce the value of a variable

END: This is the end of your pseudocode

**Conditionals**

During algorithm development, we need statements that evaluate expressions and execute instructions depending on whether the expression evaluated to True or False. Here are some common conditions used in Pseudocode:

Algoritma geliştirme sırasında, ifadelerin doğru veya yanlış olarak değerlendirilmesine bağlı olarak ifadeleri değerlendiren ve talimatları yürüten ifadelere ihtiyacımız vardır. Pseudocode'ta kullanılan bazı yaygın koşullar şunlardır:

* **IF — ELSE IF — ELSE**

This is a conditional that is used to provide statements to be executed if a certain condition is met. This also applies to multiple conditions and different variables. Here is an if statement with one condition:

Bu, belirli bir koşul karşılandığında yürütülecek ifadeler sağlamak için kullanılan bir koşuldur. Bu aynı zamanda çoklu koşullar ve farklı değişkenler için de geçerlidir. İşte bir koşulu olan bir if ifadesi:

IF you are happy

THEN smile

ENDIF

Here is an if statement with an else section. Else allows for some statements to be executed if the “if” condition is not met.

Başka bir bölüme sahip bir if ifadesi. Else, “if” koşulu karşılanmazsa bazı ifadelerin yürütülmesine izin verir.

IF you are happy

THEN smile

ELSE

frown

ENDIF

We can add additional conditions to execute different statements if met.

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**

Case structures are used if we want to compare a single variable against several conditions.

Tek bir değişkeni çeşitli koşullarla karşılaştırmak istiyorsak vaka yapıları kullanılır.

INPUT color

CASE color of

red: PRINT "red"

green: PRINT "green"

blue: PRINT "blue"

OTHERS

PRINT "Please enter a value color"

ENDCASE

The OTHERS clause with its statement is optional. Conditions are normally numbers or characters.

İfadesi ile birlikte OTHERS yan tümcesi isteğe bağlıdır. Koşullar normalde sayılar veya karakterlerdir.

**Iteration**

To iterate is to repeat a set of instructions in order to generate a sequence of outcomes. We iterate so that we can achieve a certain goal.

* **FOR structure**

The FOR loop takes a group of elements and runs the code within the loop for each element.

FOR döngüsü, bir öğe grubunu alır ve her öğe için döngü içindeki kodu çalıştırır.

FOR every month in a year

Compute number of days

ENDFOR

* **WHILE structure**

Similar to the FOR loop, the while loop is a way to repeat a block of code as long as a predefined condition remains true. Unlike the FOR loop, the while loop evaluates based on how long the condition will remain true.

FOR döngüsüne benzer şekilde while döngüsü, önceden tanımlanmış bir koşul doğru kaldığı sürece bir kod bloğunu tekrarlamanın bir yoludur. FOR döngüsünün aksine while döngüsü, koşulun ne kadar süre gerçek kalacağını değerlendirir.

To avoid a scenario where our while loop runs infinitely, we add an operation to manipulate the value within each iteration. This can be through an increment, decrement, et cetera.

While döngümüzün sonsuz olarak çalıştığı bir senaryodan kaçınmak için, her yineleme içindeki değeri değiştirmek için bir işlem ekliyoruz. Bu bir artış, eksiltme, et cetera olabilir.

PRECONDITION: variable X is equal to 1

WHILE Population < Limit

Compute Population as Population + Births — Deaths

ENDWHILE

**Functions**

When solving advanced tasks it is necessary to break down the concepts in a block of statements in different locations. This is especially true when the statements in question serve a particular purpose. To reuse this code, we create functions. We can then call these functions every-time we need them to run.

Gelişmiş görevleri çözerken, farklı konumlardaki bir ifade bloğundaki kavramları yıkmak gerekir. Bu, özellikle söz konusu ifadeler belirli bir amaca hizmet ettiğinde geçerlidir. Bu kodu yeniden kullanmak için işlevler yaratırız. Daha sonra bu işlevleri çalıştırmak için her ihtiyaç duyduğumuzda çağırabiliriz.

Function clear monitor

Pass In: nothing

Direct the operating system to clear the monitor

Pass Out: nothing

Endfunction

To emulate a function call in pseudocode, we can use the Call keyword

call: clear monitor

**Program Wrapping**

After writing several functions in our pseudocode, we find the need to wrap everything into one container. This is to improve readability and make the execution flow easier to understand.

Sözde kodumuza birkaç işlev yazdıktan sonra, her şeyi bir kaba sarma ihtiyacını buluyoruz. Bu, okunabilirliği artırmak ve yürütme akışının daha kolay anlaşılmasını sağlamak içindir.

To do this, we wrap our code as a program. A program can be defined as a set of instructions that performs a specific task when executed.

Bunu yapmak için kodumuzu bir program olarak sararız. Bir program yürütüldüğünde belirli bir görevi gerçekleştiren bir dizi talimat olarak tanımlanabilir.

PROGRAM makeacupoftea

END

**Exception Handling**

An exception is an event which occurs during program execution that disrupts the normal flow of the instructions. These are events that are non-desirable.

İstisna, programın yürütülmesi sırasında, talimatların normal akışını bozan bir olaydır. Bunlar arzu edilmeyen olaylardır.

We need to observe such events and execute code-blocks in response to them. This is called exception handling.

Bu tür olayları gözlemlemeli ve bunlara yanıt olarak kod blokları yürütmeliyiz. Buna istisna işleme denir.

BEGIN

statements

EXCEPTION

WHEN exception type

statements to handle exception

WHEN another exception type

statements to handle exception

END

**Conclusion**

There are no technical rules for Pseudocode. It is meant to be human-readable and still convey meaning and flow.

Pseudocode için teknik kural yoktur. İnsan tarafından okunabilir olması ve hala anlam ve akış iletmesi amaçlanmıştır.

There are different guide and tutorials which lean more towards language-specific pseudocode, examples of such are Fortran style pseudocode, Pascal style pseudocode, C style pseudocode and Structured Basic style pseudocode.

Dile özgü sahte kodlara daha fazla yaslanan farklı rehber ve öğreticiler vardır, bunlara örnekler Fortran tarzı sahte kod, Pascal tarzı sahte kod, C tarzı sahte kod ve Structured Basic tarzı sahte koddur.

How to Write Pseudocode - 2

Now that you know what pseudocode looks like, it’s time to learn key aspects of how to write it.

We’re going to break down the process of writing pseudocode step by step so that you know exactly how to write it and how to use it effectively.

Artık sahte kodun neye benzediğini bildiğinize göre, onu nasıl yazacağınızı öğrenmenin zamanı geldi.

Tam olarak nasıl yazılacağını ve etkili bir şekilde nasıl kullanılacağını öğrenmek için sahte kod yazma sürecini adım adım çözeceğiz.

**1. Understand the Uses**

It’s difficult to use pseudocode if you don’t actually understand its many uses.

Birçok kullanımını gerçekten anlamıyorsanız, sözde kod kullanmak zordur.

For starters, pseudocode simply makes the task of creating a new computer program more simple and straightforward.

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

Writing out the code in English enables you to create a verbal outline to follow during the programming stages of the project.

Kodun İngilizce yazılması, projenin programlama aşamalarında takip edilecek sözel bir taslak oluşturmanızı sağlar.

Pseudocode gives you the tools needed to ensure that everything you need will be included during programming. It lets you catch mistakes before they become mistakes.

Pseudocode, programlama sırasında ihtiyacınız olan her şeyin dahil edilmesini sağlamak için gereken araçları sağlar. Hataları, hata haline gelmeden yakalamanızı sağlar.

It’s also highly beneficial to use pseudocode for group projects. It breaks the program down in a simple manner so that all programmers are on the same page.

Grup projeleri için sözde kod kullanmak da oldukça faydalıdır. Programı basit bir şekilde bozar, böylece tüm programcılar aynı sayfada olur.

**2. Pseudocode is Subjective**

The trickiest thing about pseudocode is that it is subjective.

Sahte kodla ilgili en zor şey, öznel olmasıdır.

There is no standard way to write pseudocode. The goal is simply to properly outline everything in your mind.

Sahte kod yazmanın standart bir yolu yoktur. Amaç, aklınızdaki her şeyi düzgün bir şekilde özetlemektir.

That said, there are certain structures and standard procedures you should use if you’re working with others. Follow these rules to ensure that everyone else on the team is on the same page.

Bununla birlikte, başkalarıyla çalışıyorsanız kullanmanız gereken belirli yapılar ve standart prosedürler vardır. Takımdaki diğer herkesin aynı sayfada olduğundan emin olmak için bu kuralları uygulayın.

Perhaps the most important rule is to place clarity first. Make your pseudocode as clear and concise as possible, so there is no question as to what you mean.

Belki de en önemli kural önce netlik koymaktır. Sahte kodunuzu olabildiğince açık ve özlü hale getirin, bu yüzden ne demek istediğinizle ilgili bir soru yoktur.

**3. Algorithms and Basic Constructs**

Two of the most important things you must understand when it comes to writing pseudocode are algorithms and basic algorithm constructs.

Sözde kod yazarken anlamanız gereken en önemli şeylerden ikisi algoritmalar ve temel algoritma yapılarıdır.

* **Understand Algorithms –** An algorithm is the steps you must take to achieve a specific goal. Algoritma, belirli bir hedefe ulaşmak için atmanız gereken adımlardır.
* **Know Algorithm Flow –** The most basic algorithm construct or flow is “sequence,” “selection,” and “iteration.” These lay out the proper way to write the code. En temel algoritma yapısı veya akışı “dizi”, “seçim” ve “yineleme” dir. Bunlar kodu yazmak için doğru yolu gösterir.
* **Combine the Pieces –** Take the information you want to relay and use algorithm flow to create a straightforward outline. Aktarmak istediğiniz bilgileri alın ve basit bir anahat oluşturmak için algoritma akışını kullanın.

**4. Standard Procedure**

As mentioned above, there is no standard procedure for writing pseudocode. However, that doesn’t mean there aren’t certain rules you should follow.

Yukarıda belirtildiği gibi, sahte kod yazmak için standart bir prosedür yoktur. Ancak bu, izlemeniz gereken belirli kurallar olmadığı anlamına gelmez.

Follow these basic rules to ensure that everyone you’re collaborating with understands your pseudocode.

Birlikte çalıştığınız herkesin sahte kodunuzu anlamasını sağlamak için bu temel kuralları izleyin:

* **One Statement Per Line –** Express each statement or action on its own line. Her bir ifadeyi veya eylemi kendi satırında ifade edin.
* **Capitalize Directions –** Capitalize directions to highlight their importance (for example, “READ”). Önemlerini vurgulamak için yol işaretlerini büyük harflerle yazın (örneğin, “OKUYUN”).
* **Focus on Meaning –** Write what the program will do. Don’t write how to program it. Programın ne yapacağını yazın. Nasıl programlayacağınızı yazmayın.
* **Standard Programming Structures –** Follow the algorithm flow discussed above to create easy-to-follow structures. İzlemesi kolay yapılar oluşturmak için yukarıda tartışılan algoritma akışını izleyin.
* **Utilize Blocks –** Group similar actions together into blocks to separate the pseudocode into separate steps. Sözde kodu ayrı adımlara ayırmak için benzer işlemleri bloklar halinde gruplayın.

**5. Important Tips**

Once again, while pseudocode doesn’t have any hard and fast rules, there are certainly some things you should do to make understanding the pseudocode easier for all involved.

Bir kez daha, sözde kodun zor ve hızlı kuralları olmasa da, sözde kodun anlaşılmasını herkes için daha kolay hale getirmek için yapmanız gereken bazı şeyler kesinlikle vardır.

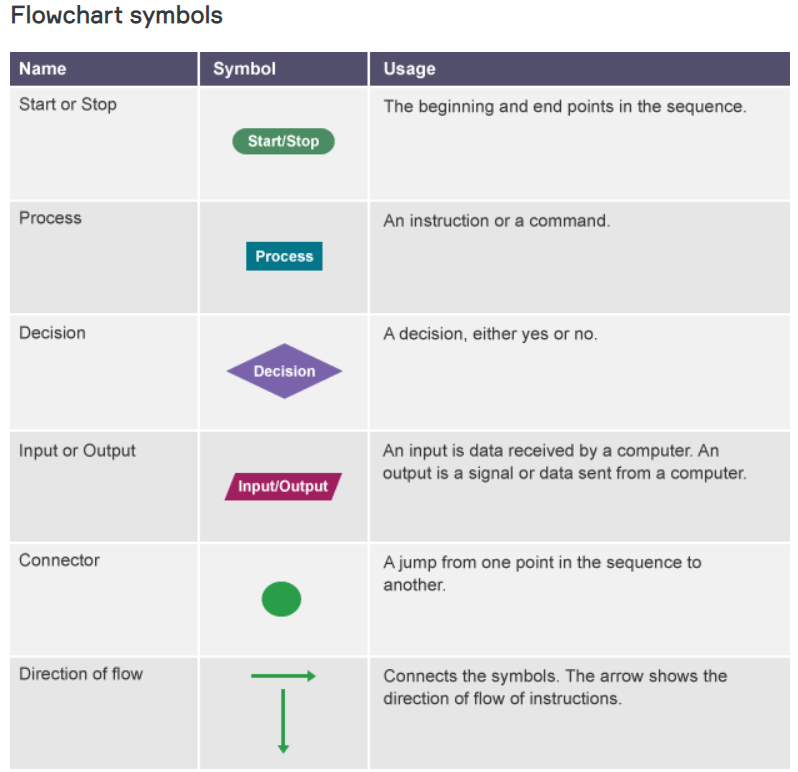
* **Keep It Simple –** Simplicity and clarity are key. Write down what the actions will be, not how to program them. Basitlik ve netlik çok önemlidir. Eylemlerin nasıl olacağını yazın, nasıl programlayacağınızı yazın.
* **Explain Everything –** Don’t include information without explaining it. Add comments to explain your steps and reasoning if needed. Bilgiyi açıklamadan dahil etmeyin. Adımlarınızı ve gerekirse muhakemenizi açıklamak için yorumlar ekleyin.
* **Practice Makes Perfect –** Just like learning a new programming language, learning how to write pseudocode takes time. Practice writing and reviewing it now. Yeni bir programlama dili öğrenmek gibi, sözde kod yazmayı öğrenmek zaman alır. Şimdi yazma ve gözden geçirme alıştırmaları yapın.
* **Review the Pseudocode –** The biggest reason to write pseudocode is to catch any mistakes before programming. So, review the finished product thoroughly to nip errors in the bud. Sözde kod yazmanın en büyük nedeni, programlamadan önce herhangi bir hatayı yakalamaktır. Bu nedenle, tomurcuktaki hataları silmek için bitmiş ürünü iyice inceleyin.
* **Translate into Programming Language –** Implement the pseudocode by tracing it with your computer language. Compare the finished product to the pseudocode. Sözde kodu bilgisayar dilinizle izleyerek uygulayın. Bitmiş ürünü sahte kodla karşılaştırın.

## Flowchart

### Representing an Algorithm: Flowchart

A flowchart is a diagram that represents a set of instructions. Flowcharts normally use standard symbols to represent different instructions. There are few real rules about the level of detail needed in a flowchart. Sometimes flowcharts are broken down into many steps to provide a lot of detail about exactly what is happening. Sometimes they are simplified so that a number of steps occur in just one step.

Akış şeması, bir dizi talimatı temsil eden bir diyagramdır. Akış şemaları normalde farklı talimatları temsil etmek için standart sembolleri kullanır. Bir akış şemasında gereken ayrıntı düzeyi hakkında birkaç gerçek kural vardır. Bazen akış şemaları, tam olarak neler olduğu hakkında çok fazla ayrıntı sağlamak için birçok adıma bölünür. Bazen basitleştirilirler, böylece sadece bir adımda birkaç adım oluşur.



A **simple** program could be created to ask someone their name and age, and to make a comment based on these. This program represented as a flowchart would look like this:

