Bölüm 2: Programlama

JAVA ile Nesne Yönelimli Programlama

Programlama Dilleri

- Bir makine (özellikle bir bilgisayar) tarafından gerçekleştirilebilecek hesaplamaları ifade etmek amacıyla tasarlanmış yapay bir dildir.
- Makinenin davranışını kontrol eden programlar oluşturmak için kullanılır.
  - Algoritmaları kesin bir şekilde ifade ederek veya
  - İnsan ile iletişim şekli olarak kullanılabilir.
- Bilgisayarlar ve diğer cihazlar üzerinde işlevsellik sağlar.
- Algoritmaların ve mantıksal süreçlerin ifadesi için kullanılır.
- Programlama dili seçimi projenin ihtiyaçlarına göre yapılır.
- Örnek diller: C, C++, **Java**, Python, Prolog, Haskell, Scala, vb.

Bilgisayar Programları Oluşturma

- Her programlama dili, bir dizi temel işlem sunar.
- Programlama dilleri, temel işlemleri ve ifadeleri oluşturmak için kullanılır.
- İfadeler, daha karmaşık işlemleri ifade etmek için bir araya getirilir.
- Programlama dili, hesaplamaların veya ifadelerin sonuçlarını çıkarmak için kullanılır. Örneğin, Java'da veri türleri ve değişkenler kullanarak bu işlem gerçekleştirilir.

Dilin Yönleri

- Temel yapılar
  - Programlama dilleri ve doğal diller birçok ortak yön paylaşır.
    - **Programlama dili** - sayılar, diziler, basit operatörler
    - **Türkçe** - kelimeler
- Sözdizimi
  - Dilin kurallarını belirler ve bu, dilin anlaşılabilirliğini etkiler.
  - **Programlama dili** - Hangi karakter ve sembol dizilerinin düzgün olduğunu belirler
  - **Türkçe** - "kedi köpek çocuk" kabul edilebilir bir cümle formunda olmadığı için sözdizimsel olarak geçerli değildir

Dilin Yönleri – Statik Anlam (Semantic)

- Sözdizim açısından doğru ifadelerin anlamsal olarak doğru olup olmadığını kontrol eder.
- **İngilizce** - "I are big," form olarak <isim> <fiil> <isim> şeklinde geçerli bir sözdizimine sahip olabilir, ancak "I" tekil, "are" çoğuldur, bu nedenle dil açısından geçerli değildir.
- **Programlama dili** - <değer> <operatör> <değer> geçerli bir sözdizimsel formdur, ancak 2.3/'abc' bir statik anlamsal hata oluşturur.

Dilin Yönleri – Anlam (Semantic)

- Semantik, bir dilde sözdizim açısından doğru ifadelerin ne anlama geldiğini belirler.
  - İngilizce gibi doğal dillerde bazen çokanlamlılık olabilir.
    - “They saw the man with the telescope.”
- Programlama dilleri açısından, herhangi bir geçerli ifade tam olarak bir anlama sahiptir, ancak bu anlam programcının beklediği şey olmayabilir. Örneğin, yanlış bir işlem sonucu beklenmeyen bir sonuç verebilir.

Neler Yanlış Gidebilir?

- Sözdizimsel hatalar,
  - Yazım hataları gibi açıkça tanımlanmış kurallara aykırı durumları ifade eder ve genellikle bilgisayarlar tarafından kolayca tespit edilir.
- Statik anlamsal hatalar,
  - Dil bazı kontrolleri çalıştırmadan önce yapar. Eğer bu hatalar yakalanmazsa, programın davranışı tahmin edilemez hale gelebilir.
- Programlar, geçerli sözdizimi ve statik anlamsal hatalara sahip olsa bile, beklenen sonucu üretmeyebilir.
  - Bu durumlar, çökmeler, sonsuz döngüler veya istenmeyen sonuçlar şeklinde ortaya çıkabilir.

Amacımız

- Bir programlama dilinin sözdizim (syntax) ve anlamını (semantic) öğrenmek:
  - Bir programlama dilinin temel yapı taşlarını kavramak.
  - Sözdizim ve semantiği öğrenmek.
- Bu unsurları kullanarak sorunları çözmek için "tarifleri" bilgisayarın anlayacağı şekilde çevirmeyi öğrenmek:
  - Sorunları çözmek için bilgisayarın kullanabileceği bir formüle dönüştürmek.
- Hesaplamalı Düşünme ile Problemleri Çözme:
  - Problemleri çözmek için bir dizi yöntemi kullanabilme yeteneği.

Programlama Dilleri

- Programlama dilleri, düşük seviyeden yüksek seviyeye kadar farklı soyutlama seviyelerine sahiptir.
- **Düşük seviye diller** (örneğin Assembly), bilgisayarın neredeyse makine seviyesinde çalıştığı çok basit hesaplama talimatlarına sahiptir.
- **Yüksek seviye diller** (örneğin Python, C, Java), daha zengin ve karmaşık bir talimat kümesine sahiptir ve daha yüksek soyutlama seviyelerini destekler.

Programlama Dilleri

- Programlama dilleri, uygulama alanına göre genel amaçlı veya hedeflenmiş (targeted) olabilir.
- **Genel amaçlı diller**, çeşitli uygulamaları desteklemek için geniş bir temel talimat kümesine sahiptir.
- **Hedeflenmiş diller**, belirli bir uygulama türüne odaklanır ve bu tür işlemleri kolaylaştırmak için özelleştirilir.
- Örneğin, MATLAB sayısal hesaplamalar için özel olarak tasarlanmıştır ve matris ve vektör işlemleri gibi belirli hesaplama görevleri için optimize edilmiştir.

Programlama Dilleri

- Programlama dilleri, kaynak kodun nasıl yürütüldüğüne göre yorumlanan veya derlenen olarak sınıflandırılabilir.
- **Yorumlanan dillerde**, kaynak kod doğrudan yorumlayıcı tarafından çalıştırılır ve program akışı talimatları sırayla işler.
- **Derlenen dillerde** ise kaynak kod, yürütülmeden önce derleyici tarafından nesne koduna çevrilmelidir. Derleme işlemi, hızlı ve optimize edilmiş bir yürütme sağlayabilir.

Programlama Dil Paradigmaları

- Programlama dilleri, farklı paradigmalara dayanabilir, yani farklı yaklaşımlarla sorunları çözmek için tasarlanmıştır.
- **Fonksiyonel** diller, hesaplamayı matematiksel işlevlerin değerlendirmesi olarak ele alır. Örnekler: Lisp, Scheme, Haskell, vb.
- **Emperatif** diller, hesaplamayı program durumunu değiştiren ifadelerin terimleriyle açıklar. Örnekler: FORTRAN, BASIC, Pascal, C, vb..
- **Mantıksal** diller, hesaplamaları mantığını açıklarken kontrol akışı hakkında ayrıntı vermez. Örnek: Prolog
- **Nesne yönelimli** diller, veriyi ve işlevleri nesneler olarak düşünerek programları tasarlar ve yönetir. "Nesneleri" kullanır - veri alanları ve yöntemlerden oluşan veri yapılarıyla birlikte etkileşimlerini tasarlamak için kullanılır. Örnekler: C++, Java, C#, Python, vb.

İfadeler

- Programlama dili içinde hesaplamaları tanımlayan birer matematiksel ifadelerdir.
- Programlama dilleri, ifadeleri değerlendirmek ve sonuçlarını üretmek için belirli kuralları takip eder.
- Programlama dili, bir ifade içindeki işleçlerin sırasını ve önceliğini belirlemek için belirli kuralları izler.
  - Örneğin, çarpma ve bölme işlemleri toplama ve çıkarmadan önce gerçekleştirilir.
- İşlem öncelikleriyle ilgili olarak, üs alma operatörü **, diğerlerinden farklı olarak sağa doğru işlenir.
- İşlem önceliklerini geçersiz kılmak için parantezler kullanılır.

Örnek İfadeler

- 5
- 3+4
- 44/2
- 2**3
- 34+56
- (72 – 32) / 9 * 5

Değişkenler

- Değişkenler, programda değerleri saklamak ve yönetmek için kullanılır. Değişkenler, program içinde verilere referans oluşturur.
- Bir değişkeni atamak için, "değişkenAdı = ifade" kullanılır. Örneğin, "pi = 3.14" pi değişkenine 3.14 değerini atar.
- Değişken adları bazı kurallara uymalıdır:
  - Değişken adları sadece bir kelime olmalıdır (boşluk içermez).
  - Değişken adları yalnızca harf, rakam ve alt çizgi (_) içerebilir.
  - Değişken adları bir rakamla başlayamaz.

Değişken Tanımlama

- **Dinamik tipli** dillerde,
  - değişkenlerin türünü önceden belirlemeye gerek yoktur. Değişken kullanıldığında türü otomatik olarak belirlenir.
  - Örneğin, Python gibi dinamik tipli dillerde, bir değişkenin türü çalışma zamanında değiştirilebilir. Bu, esneklik sağlar, ancak hataların çalışma zamanında ortaya çıkma olasılığını artırabilir.
- **Statik tipli** dillerde,
  - değişken türü açıkça bildirilmelidir ve karakter dizisi ile tamsayı gibi uyumsuz işlemler derleme zamanında hata olarak tespit edilir.

Atama İşlemi

- Bir değişkenin değerini değiştirmek için kullanılır.
- Atama işlemi iki adımda gerçekleşir:
  - Sağ taraf ifadesi değerlendirilir. Bu, ifadenin hesaplanması anlamına gelir. Örneğin, bir aritmetik işlem veya bir fonksiyon çağrısı olabilir.
  - Hesaplanan değer, atama operatörü tarafından belirtilen değişkene saklanır. Değişkenin içeriği bu yeni değere güncellenir.
- Örneğin, x = 5 ifadesinde, sağ taraftaki 5 ifadesi değerlendirilir ve bu değer x adlı değişkende saklanır. Bu, x değişkeninin değerini 5 yapar.
- Atama işlemi, programın değişkenlerin içeriğini güncellemesini sağlar ve verilerin işlenmesi ve saklanmasında temel bir rol oynar.

Koşullu İfadeler

- Programın hangi yolu izleyeceğini belirlemek için kullanılır. İfadenin sonucu doğru (True) veya yanlış (False) olabilir.
- Atama işlemi (örneğin, x = 100), bir koşul ifadesi değildir. Bu ifade, bir değeri bir değişkene atamak için kullanılır.
- **Karşılaştırma operatörleri**, değerleri karşılaştırmak ve koşullu ifadeler oluşturmak için kullanılır. Örneğin, x >= 5, x'in 5 veya daha büyük olup olmadığını kontrol eder.
- **Boolean operatörler**, koşulları birleştirmek veya tersine çevirmek için kullanılır. Örneğin, not True ifadesi True'ın tersidir.
- **Karma operatörler**, sayıları veya ifadeleri karşılaştırmak için kullanılır. Örneğin, 3 < 4 and 5 < 6, hem 3 < 4 doğru hem de 5 < 6 doğru olduğunda doğru sonuç verir.

Dizgeler (Strings)

- Dizgeler (strings), metin verilerini temsil etmek için kullanılır. Örneklerde «Java» veya "BBM 101 - Programlamaya Giriş" gibi metinler bulunabilir.
- Boş bir dizge (""), bir atama yapılmamış değişkenle aynı değildir. Her ikisi de farklı değerlere sahiptir.
- Kaçış (escape) dizileri, özel karakterleri temsil etmek için kullanılır. Örneğin, '\t' sekme karakterini, '\n' yeni satırı temsil eder.
- Dizge işlemleri, uzunluğu bulma, birleştirme ve belirli karakterleri arama gibi işlemleri içerir.
  - Örneğin, 'a' karakterinin dize içinde olup olmadığını kontrol edebiliriz.

Farklı Türler Karşılaştırılmamalıdır

- Farklı veri türlerini karşılaştırmak hatalara neden olabilir. Örneğin, bir tamsayı ile bir dizgeyi doğrudan karşılaştırmak hata verebilir.
- Programlama dilleri, veri türlerinin karşılaştırılması için uygun kuralları ve tür dönüşümünü tanımlar. Ancak, farklı türler arasında karşılaştırma yaparken dikkatli olunmalıdır.
- Veri türlerinin uygun şekilde karşılaştırılması, programın doğru çalışmasını ve beklenmeyen hataların önlenmesini sağlar.

Farklı Türler Üzerinde İşlemler Farklı Davranır

- İşlemler, farklı veri türleri üzerinde farklı şekilde davranır. Örneğin, dizgeler arasındaki toplama işlemi birleştirme (concatenation) olarak çalışır.
- Farklı türler arasında işlem yapmak hatalara neden olabilir. Örneğin, bir sayıyı bir dizge ile toplamak veya bir sayıyı mantıksal bir değerle toplamak
- 3.0 + 4.0 # Doğru
- 3 + 4 # Doğru
- 3 + 4.0 # ???
- "3" + "4" # Birleştirme (Concatenation)
- 3 + "4" # Hata
- 3 + True # Hata

Program Bir Tariftir (A Program is a Recipe)

- Programlama, bir sorunu çözme sürecidir ve bir program, bu sürecin sonucunda ortaya çıkar.
- Bir program, bilgisayara belirli bir görevi nasıl yerine getireceğini anlatan bir tariftir.
- Programlar, bilgisayarlar tarafından anlaşılabilir bir şekilde yazılır. Adım adım talimatlar içerir.
- Bilgisayarlar, talimatlara uyarak işlemleri gerçekleştirir.
- Örnek:
  - Bir e-posta uygulaması, e-posta gönderme işlemi için bir programdır.
  - Bir oyun, oyunun nasıl oynanacağını belirleyen bir program içerir.

Kodlamadan Önce Algoritmayı Tasarlayın

- Bir programı yazmadan önce, sorunu nasıl çözeceğinizi düşünmek ve bir algoritma tasarlamak önemlidir. Algoritma, mantıklı bir çözüm yolunu ifade eder.
- Algoritmik düşünme, bir problemi çözme ve işlemleri sıralama yeteneğidir. İyi bir algoritma, programın verimli bir şekilde çalışmasına yardımcı olur.
- Kodlama, algoritmayı bilgisayarın anlayabileceği sözdizimine çevirme işlemidir. Kodlama, tasarlanan algoritmayı gerçek bir program haline getirir.
- Algoritma tasarlama, programlama görevlerinin ayrılmaz bir parçasıdır ve bu süreci bir alışkanlık haline getirmek, daha etkili bir programcı olmanıza yardımcı olur.

Bir Program Nedir?

- Bir program, bir dizi talimatı içerir. Bu talimatlar, bilgisayar tarafından birer birer işlenir ve sonuç olarak belirli bir görev gerçekleştirilir.
- Bilgisayar, programın talimatlarını sırayla takip eder ve bu nedenle programların işlem sırasını belirlemek önemlidir.
- Çalışmalarınızı bir program olarak kaydetmek, tekrar kullanmak veya paylaşmak istediğinizde zamandan tasarruf etmenizi sağlar. Bu nedenle, programlama çalışmalarınızı programlar olarak kaydetmek iyi bir uygulamadır.

İfadeler, Deyimler ve Programlar

- Bir ifade, bir değeri hesaplar ve sonucunu döndürür. Örneğin, 3 + 4 ifadesi 7 değerini hesaplar.
- Bir deyim bir etki yaratabilir. Örneğin, pi = 3.14159 ifadesi, pi adlı değişkene bir değer atar ve print(pi) ifadesi, ekrana pi değerini yazar.
- İfadeler, başka ifadelerin ve deyimlerin içinde görünebilir. Bu, daha karmaşık hesaplamalar ve işlemler yapmamıza olanak tanır.
- Bir deyim bir ifadenin içinde görünemez. Örneğin, 3 + print(pi) ifadesi hata verir, çünkü print ifadesi bir etki yaratır, ancak bir değer döndürmez.
- Bir program, birçok deyim içerir.
- Bir program bir şey yapmalı veya bilgi iletmelidir.

Popüler Programlama Dilleri

- Programlama dilleri, farklı görevler için farklı avantajlar sunar. Programcılar, projelerinin ihtiyaçlarına en uygun dili seçerler.
- Her yıl, programlama topluluğu tarafından kullanılan programlama dillerinin popülerliği değerlendirilir ve en iyi diller belirlenir.
- 2023'ün en öne çıkan dilleri arasında Python, **Java**, JavaScript, C#, C++, Rust, Swift, Kotlin ve Go (Golang) gibi diller bulunur.

Öne Çıkan Diller

- **Python**: Veri analizi, yapay zeka, web geliştirme gibi alanlarda yaygın
- **Java**: Büyük ölçekli uygulamalar ve Android uygulamaları için popüler
- **JavaScript**: Web tarayıcıları için önemli, web geliştirme için uygun
- **C#**: Windows uygulamaları ve oyun geliştirmek için kullanılır
- **C++**: Oyun, sistem programlama, performans odaklı uygulamalar için
- **Rust**: Güvenli ve hızlı sistem programlama için kullanılır
- **Swift**: Apple platformları üzerine uygulama geliştirmek için kullanılır
- **Kotlin**: Android uygulama geliştirme için popüler
- **Go** (**Golang**): Basit ve etkili bir dil, ağ uygulamaları için ideal

SON