

**T.C.**

**YALOVA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**PROJE ÖDEVİ**

**GO PROGRAMLAMA DİLİ**

**HAZIRLAYANLAR**

**150101053 MUSA KAVUŞ**

**DANIŞMAN: MUHAMMED TEKİN**

**YALOVA-2021**

# Önsöz

Bilgisayar programlama, bilgisayarların nasıl çalıştığını tanımlayan sanat ve bilimdir. Bu proje sizlere Google tarafından tasarlanan Go programlama dilini kullanarak bilgisayar programlarının nasıl yazılacağını öğretecektir. Go, gelişmiş özelliklere ve temiz bir söz dizimine sahip genel amaçlı bir programlama dilidir. Çeşitli platformlarda geniş kullanılabilirliği, sağlam ve iyi belgelenmiş ortak kitaplığı ve iyi yazılım mühendisliği ilkelerine dayanması nedeniyle Go, programlama dillerine başlangıç olarak ideal bir dildir. Go ile yazılım yazmak için kullandığımız süreç oldukça basittir:

\* İhtiyaçları toplayın.

\* Çözüm bulun.

\* Çözümü uygulamak için kaynak kodu yazın.

\* Kaynak kodu yürütülebilir bir dosyada derleyin.

\* Çalıştığından emin olmak için programı çalıştırıp test edin.

# ****Go Programlama Dili Temel Bilgiler****

## 2.1. Hakkında

Pragmatik ve minimalist bir dildir. Fayda ve basitlik odaklı olduğundan daha önce bahsetmiştik. Go'da sadece gerekli ve işlevsel olan eşyaları yanınıza alarak verimli uygulama geliştirebilirsiniz. Yazım biçimi C diline benzer, öğrenmesi ve kod okuması kolaydır. Statik tiplidir, gerekirse dinamik tip de kullanılabilir. Bir dilin statik veya dinamik tipli olması, dilin karakteristiğini belirleyen önemli bir farktır. Statik dillerde, derleme sırasında (compile time) değişken tip kontrolü yapıldığından, çalışma sırasında (run-time) bu işlem için ayrıca emek harcanmaz ve performansı artırır. İki türün de birbirine göre olumlu ve olumsuz yönleri vardır. Dinamik tipli diller ise programcıya birtakım kolaylıklar sağlar. Ne türde verinin geleceği bilinmeyen durumlarda, dinamik tipli bir değişken kullanımı pratik olmaktadır. Ancak dinamik tip kullanımının sorun giderme (debug) durumunda olumsuz etkisi olabilmektedir. Bir kod yığınının içinde bir değişkenin tipinin ne olduğu o andaki değerine bağlıdır. Bazen de değişkene farklı tipte veri gelmesi nedeniyle programda analizi güç olan hatalar oluşabilmektedir. Go'da bu nedenle statik tip kullanımı tercih edilmiştir. Ancak istendiğinde dinamik değişken kullanma esnekliği de bulunmaktadır.

Doğal çok görevlilik kabiliyeti vardır. Günümüz bilgisayar sistemlerinde, işlemcilerde aynı anda çok sayıda işlem yapmak için çok sayıda çekirdek bulunmaktadır. Ancak programlama dillerinde çok görevli işlemler oluşturmak genellikle kolay değildir. Bazı dillerin bu konuda desteği yoktur. Çok görevli çalışan program yazma kabiliyeti olan dillerde de bu özellik genellikle ilave kütüphanelerle eklenmekte ve kodu karmaşıklaştırmaktadır. Oysa Go'nun amacı günümüz problemlerine gerçek çözümler getirmek olduğundan, doğal olarak çok görevlilik yeteneği ile tasarlanmıştır. Go dilinde bu yeteneğe concurrency (eş zamanlılık) denmektedir. Go, eş zamanlılığı sağlayabilmek için, goroutine ve channel ismi verilen özellikleri kullanır.

Derleme işlemleri çok hızlıdır. C ve C++ dillerinde büyük projeler geliştirenlerin en büyük şikâyeti, derleme süresinin uzun olmasıdır. Performans, Go’nun öncelikli hedeflerinden olduğu için, derleme süresi kısa olacak şekilde tasarlanmıştır.

Büyük işler ve büyük yazılım ekipleri için tasarlanmıştır. Kodların kolay okunabilir olması, Go için olmazsa olmazdır. Parantezler, boşluklar olması gereken yerde olmadığında hata vermektedir. Bu nedenle, otomatik kod düzenleme sistemi (GoFmt) hazır gelmektedir. Bir Go kodunu kim yazarsa yazsın, aynı tarzda görünecektir.

Genel amaçlıdır. Her programlama dili, kendi çağının ihtiyaçları için tasarlanmıştır. Örneğin; PHP’nin açılımı kişisel ev sayfaları şeklide olmasına rağmen, PHP ile dev projeler yapılmıştır. Ayrıca PHP ilk çıktığı günden itibaren sürekli yenilenerek güncel ihtiyaçlara cevap verme konusundan popüler dillerden birisi olmuştur. C ve C++ gibi diller genellikle performans odaklı projeler ve sistem programlama için kullanılmaktadır. Python ve Ruby gibi diller; hızlı ve kolay uygulama geliştirme odaklıdır. Objective-C ve Swift dilleri, sadece mobil programlama aracıdır. NodeJS, sunucu tarafında JavaScript kullanımı ile popüler olmuştur ancak performans odaklı uygulamalarda çok başarılı değildir. Go ve Java dilleri bu açıdan diğerlerinden ayrı bir yere sahiptir. İkisi de genel amaçlı kullanımı hedeflemektedir.

### 2.1.1. Diğer Diller ile Karşılaştırma

Çıkış noktası günümüzün büyük problemlerine gerçek ve kolay çözümler üretmek olan bir dilin, diğer dillerle kıyaslanması çok kolay değil aslında. En çok kıyaslanan dillere göre bazı farklarını şu şekilde sıralayabiliriz:

• Node.JS gibi belirli bir konuya odaklanmış değildir.

• C++ gibi derlemesi uzun süren ve zor taşınabilir bir dil değildir.

• Java gibi kompleks değildir.

• Ruby ve Python gibi dinamik tipli ve yorumlanan bir dil değildir.

Yukarıdakiler gibi çok sayıda madde yazılabilir. Yüksek bir arazi aracı ile, sürat yapmak için tasarlanmış alçak bir spor otomobili karşılaştırmak çok anlamlı olmayacaktır. Benzer şekilde; Go'nun diğer dillerle yapılan karşılaştırmalarında bir şeyler mutlaka eksik kalacaktır.

2.2. Go’nun Yapısal Açıdan Programlama Dilleri Arasındaki Yeri

Yapısal anlamda, Go'nun yerini (ve diğer dillerden farklı yönlerini) anlayabilmek için, aşağıdaki çizimi hazırladık. Programlama dillerinin çoğu bu çizim üzerinde bir yere hemen oturabilir. Ancak Go'nun bu sınıflandırmaya hemen yerleşmesi pek kolay değil.

Daha önce de belirttiğimiz gibi, Go pragmatiktir; verimlilik, performans ve kolaylık söz konusu olduğunda Go, geleneklere bağlı kalmamaktadır.

### 2.1.2 Go’nun Yapısal Açıdan Programlama Dilleri Arasındaki Yeri

Yapısal anlamda, Go'nun yerini (ve diğer dillerden farklı yönlerini) anlayabilmek için, aşağıdaki çizimi hazırladık. Programlama dillerinin çoğu bu çizim üzerinde bir yere hemen oturabilir. Ancak Go'nun bu sınıflandırmaya hemen yerleşmesi pek kolay değil. Daha önce de belirttiğimiz gibi, Go pragmatiktir; verimlilik, performans ve kolaylık söz konusu olduğunda Go, geleneklere bağlı kalmamaktadır.

### 2.1.3. Go'nun Kullanım Amacı

Go'nun genel amaçlı bir dil olduğundan daha önce bahsetmiştik. Özellikle yüksek performans gerektiren sistemler için oldukça avantajlıdır. Hızlı ve pratik çözüm geliştirme konusunda iddaalıdır. Sunucu taraflı uygulamalar ve kurumsal uygulamalar alanında yine ciddi bir oyuncudur. Android ve Apple iOS için doğal (native) uygulama geliştirilebilmektedir.

Go Ekosistemi (işletim araçları, standart kütüphane, üçüncü parti kütüphaneler) çok farklı türde uygulamalar için oldukça geniş olanaklar sunmaktadır. Web programlama konusunda hem kullanıcı tarafında hem de sunucu tarafında kullanılabilmekte ve güzel uygulamalar yapılabilmektedir. Çok sayıda hazır web framework (web programlama çatısı) seçeneği bulunmaktadır. Bu framework'ler ile hazır MVC mimarisi kullanılabilmekte, hazır ORM katmanı sayesinde farklı veritabanlarına kolayca bağlantı yapılabilmektedir.

Tasarım aşamasında C++ dilinden etkilendiğinden, sistem programlama için de çok uygun bir dildir. Raspberry'nin GPIO pinlerini doğrudan kullanabilmek için kütüphane kullanıma hazır durumdadır. Benzer şekilde Arduino, Beaglebone, Intel Edison, Pebble vb. birçok donanım için kütüphaneler kullanımdadır.

Bilimsel çalışmalar için de çok sayıda kütüphane ve örnek bulunuyor. Doğal dil işleme, yapay sinir ağları, örüntü tanıma, sınıflandırma, genetik algoritmalar, karar ağacı, dağıtık veri işleme, derin öğrenme, veri görselleştirme, istatistik vb. birçok farklı alanda kütüphaneler ile performanslı çalışan uygulamaları kolayca geliştirmek mümkündür.

## 2.2. TARİHÇESİ

İlk kez 2007 yılında Google'da Robert Griesemer, Rob Pike ve Ken Thompson tarafından kısmi zamanlı çalışılarak başlatılmıştır. Projenin çıkış amacı: C++ ve Java gibi genel amaçlı ve ölçeklenebilir, ancak bu diller kadar karmaşık olmayan, günümüz problemlerine kolay ve hızlı çözüm üretebilecek bir programlama dili tasarlamak şeklindedir.

Proje daha sonra tam zamanlı çalışılan bir projeye dönüştürülmüş, 2009 yılında da açık kaynak olarak duyurusu yapılmıştır. Sonrasında pek çok gönüllü tarafından projeye katkıda bulunulmuştur. 2012 yılında ise ilk kararlı sürüm yayınlanmıştır. Rob Pike'ın 2012 yılında yaptığı bir konuşmada Go'nun çıkışı şu şekilde ifade ediliyor: "Go; Google tarafından, Google'ın problemlerini çözmek için tasarlandı ve Google'ın zorlu problemleri var". Go resmî web sitesinde projenin odakları şu şekilde sıralanmıştır: **performans, güvenilirlik, taşınabilirlik, gerekli olan yeni özelliklerin eklenmesi.**

* Statik yazılmış, büyük sistemlere ölçeklenebilir olması (**Java** ve **C++** gibi)
* Üretken ve okunabilir olması, çok fazla zorunlu anahtar kelime ve tekrarlamaların kullanılmaması[[7]](https://tr.wikipedia.org/wiki/Go_(programlama_dili)#cite_note-7)
* **Tümleşik geliştirme ortamına** ihtiyaç duymaması ancak desteklemesi
* Ağ (networking) ve çoklu işlemleri (multiprocessing) desteklemesi

Yapılan röportajlarda bu üç dil tasarımcısının yeni dili yapmalarındaki motivasyonun, **C++**'daki karmaşıklığı beğenmedikleri için geldiğini belirtmişlerdir.

Go 1.0, Mart 2012'de yayınlanmıştır. En son Ağustos 2018'de Go 1.11 sürümü yayınlanmıştır.

Kasım 2016'da Go ve Go Mono yazı tipleri, özellikle Go projesi tarafından kullanılmak üzere yazı tasarımcıları Charles Bigelow ve Kris Holmes tarafından piyasaya sürüldü. Go, Lucida Grande'ye benzeyen hümanist bir sans-seriftir ve Go Mono tek aralıklıdır. Yazı tiplerinin her biri WGL4 karakter kümesine uygundur ve büyük bir x yüksekliği ve farklı harf biçimleriyle okunaklı olacak şekilde tasarlanmıştır. Hem Go hem de Go Mono, eğik çizgili sıfır, kuyruklu küçük l harfi ve serifli büyük I harfi ile DIN 1450 standardına uygundur. Nisan 2018'de, orijinal logo, sağa doğru eğimli, sondaki akış çizgileriyle stilize edilmiş bir GO ile değiştirildi. Ancak, Gopher maskotu aynı kaldı.

## 2.3 Yapılmış Projeler ve Kullanım Alanları

* **DOCKER** platformu Go ile geliştirildi."Geliştirdiğim uygulama benim bilgisayarımda çalışıyordu, sunucuya attım hata verdi" tarzındaki sorunların tümünü ortadan kaldıran müthiş sanallaştırma teknolojisidir. Kendi bilgisayarınızda uygulamayı bir Docker konteynır içinde geliştirirsiniz. Bu konteynır nereye (sunucuya, buluta, arkadaşınıza vb.) giderse, tüm bileşenleri (bağımlılıklar, kütüphaneler, konfigürasyonlar vb.) ile bütün olarak gider. Ayrıca bir konteynır çalışırken, klasik işletim sistemi sanallaştırma gibi kaynak tüketmez. Mevcut işletim sisteminin üzerinde uygulama sanallaştırıyormuş gibi çalışır.
* **KUBERNATES**, Google tarafından Go ile ve açık kaynaklı olarak geliştirildi. Bu platform da çok sayıda sunucuyu kümeleyerek, çok-çok-çok sayıda Docker konteynırını bu küme yapısı üzerinde esnek bir şekilde çalıştırmak için kullanılıyor. Microsoft, IBM, RedHat gibi devler de kubernetes'i destekliyor.
* **NTP Pool**, bilgisayar sistemlerinin saatini senkronize etmek için hizmet veriyor. Her saniye binlerce sorgu alan ve Perl ile geliştirilmiş olan bu sistem performans sorunu nedeniyle 2012 yılında Go ile yeniden yazıldı.
* **GOOGLE**’ın download sitesi, C++ ile geliştirilmişti. Sürekli değişikliklerle hantallaşan ve yavaş hizmet vermeye başlayan bir sistem olması nedeniyle yeniden Go ile yazıldı.
* **CERN**'de sunucu tarafında DAS (Data Access Service ~ Veri Erişim Hizmeti) Python'dan Go diline geçirildi. Toplam kod miktarı 6 kat azaldı. Kullanılan bellek miktarı 1,5-2 kat azaldı. İstemcilere cevap verme süresi 10 kat hızlandı. Internet üzerinde çok sayıda başarı öyküsü bulunmaktadır.

Go dilinin birincil amacı sistem programlamalarının yapılmasıdır. Aktif biçimde sunucu tarafında kullanılması için geliştirilen Go’yu, sunucu ve alt sistemler geliştirmek için harika bir dil olarak görebiliriz. Bunun yanı sıra Go ile [web](https://wmaraci.com/nedir/web) namına hızlı çalışan ve verimli projeler de üretilebilir. Henüz kısıtlı örneklerle karşımıza çıkıyor olmasına rağmen Google’ın, Go’yu [mobil](https://wmaraci.com/nedir/mobil) işletim sistemi Android’de kullanma planı vardır. Gelecekte Go’yu bilen geliştiriciler sadece sunucu ve sistem tarafındaki işleri halletmekle kalmayarak, kendi [Android](https://wmaraci.com/nedir/android) mobil uygulamalarını da Go ile verimli biçimde geliştirebilecekler.

Go’nun büyük bir kitle tarafından kabul görmesi uzun sürmemiştir. Dünya devleri olarak kabul edebileceğimiz **Docker**, **Koding**, **Google**, [Apple](https://wmaraci.com/nedir/apple), [Twitter](https://wmaraci.com/nedir/twitter), [Amazon](https://wmaraci.com/nedir/amazon), [Github](https://wmaraci.com/nedir/github), [Ubuntu](https://wmaraci.com/nedir/ubuntu), [Dropbox](https://wmaraci.com/nedir/dropbox), [Facebook](https://wmaraci.com/nedir/facebook) ve **Medium** başta olmak üzere birçok şirket Go’yu farklı ihtiyaçları için kullanıyor.

## 2.4. Dokümanlar, Kaynaklar, Topluluk Siteleri

* <https://golang.org>

Go programlama dilini Microsoft Windows, Apple ve Linux işletim sistemlerine buradan güvenli bir şekilde indirebiliriz. Go dilinde yazdığımız kodları çalıştırabilir. Forumlarında dilediğimiz soruları sorabileceğimiz bir ortam sağlamaktadır.

* <https://stackoverflow.blog/2020/11/02/go-golang-learn-fast-programming-languages/>

Stackoverflow sitesi sayesinde Go diline hızlıca bir giriş yapıp, neden Go programlama dilini öğrenmememiz gerektiği hakkında yazılar okuyabilirsiniz. Birkaç farklı Go dilinde yazılan örnekleri inceleyebilirsiniz.

* <https://www.udemy.com/course/go-the-complete-developers-guide/>

Ve tabiki de Udemy sayesinde sıfırdan uzman derecesinde bir Go bilgisini satın alıp video desteği ile detaylı bir şekilde Go dilini öğrenebilirsiniz.

Go Dili çıktığı ilk günden beri hatrı sayılır bilgisayar dehalarının dikkatini çekmiştir. Bunlardan bazıları;

* İşlevsel dillerde oldukça fazla kod ve oldukça fazla Go yazan biri olarak, ne kadar çok Go yazarsam, ilk başta dehşete düştüğüm dil özelliklerini (veya eksikliklerini) o kadar az önemsediğimi görüyorum ve diğer dillerin çoğunu aşırı karmaşık olarak görüyorum. Go teoride pek iyi bir dil değil, ama pratikte harika bir dil ve umurumda olan tek şey pratik, bu yüzden onu biraz seviyorum.  *(Supersillyus, 2011)*
* Go'dan sonra, başka herhangi bir şeyde programlama yapmak, California Eyaleti bütçesini dengelemek kadar zor görünüyor. *(Charles Thompson, 2017)*
* Go'da kod tam olarak sayfada ne yazıyorsa onu yapar.  *(Andrew Gerrand, 2015)*
* Neden teorik olarak heyecan verici olmayan bir diliniz olsun ki? Çünkü çok kullanışlı (Rob Pike, 2014)

## 2.5. Derleyici Nasıl Kurulur

Go kurulumu için farklı yöntemler bulunmaktadır. Bunların en kolayı, Go geliştiricileri tarafından internet üzerinden indirmeye sunulan ve her işletim sistemi için ayn olarak tasarlanmış kurulum (binary) dosyalarıdır. Mazoşistler, maceracılar (ve tabiiki uzmanlar) için kaynak kodları internet üzerinden indirip derleme olanağı bulunmaktadır.

Bulut ve sanallaştırma çağında olduğumuzdan, artık herhangi bir şeyi denemek için bilgisayarımıza kurmak zorunda da değiliz tabii ki. Go'yu bilgisayarına herhangi bir şey kurmadan denemek isteyenler için bazı yöntemleri hızlıca sıralayalım:

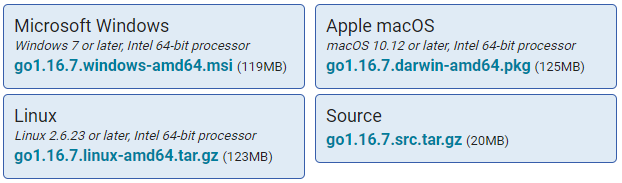
* Basit denemeler yapmak ve yazım tarzını görmek isterseniz, Go Playground'u Web sayfası üzerinden kullanabilirsiniz.
* Bilgisayarınızda bir sanallaştırma platformu (en popüler ücretsiz platformların

Oracle VirtualBox ve VMware Workstation Player olduğunu belirtelim) kurup Go geliştirme ortamı yüklü ücretsiz bir sanal makine imajı (arama motoruna Golang Virtual Appliance yazabilirsiniz) indirip deneyebilirsiniz.

* Web üzerinden çevrimiçi geliştirme ortamı hizmeti veren sitelerden faydalanabilirsiniz. Koding (koding.com), Cloud9 (c9.io), Codenvy (codenvy.com) gibi web siteleri, Go geliştirme ortamını (belirli kısıtlarla) web üzerinden ücretsiz sunuyorlar. İstenirse ücretli abonelik ve hizmet de alınabiliyor. Sadece bir web tarayıcısı kullanarak birkaç dakika içinde Go geliştirme ortamını kullanmaya başlayabilirsiniz.
* Kendinizi biraz daha uzman hissediyorsanız, sanal makine üzerinden geliştirme ortamı sağlayan platformları (Docker ve Vagrant hem ücretsiz hem de çok güçlü) inceleyebilirsiniz. İnternet'te her iki platformda kullanılabilecek çok sayıda hazır imaj bulunmaktadır.
* Google Application Engine (GAE), Amazon Web Services (AWS), Heroku benzeri bulut platformları üzerinde Go geliştirme olanaklarını inceleyebilirsiniz.

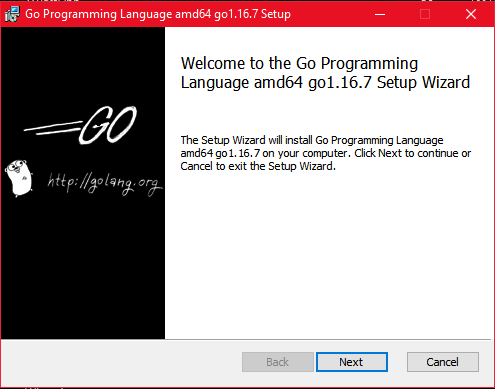
**WINDOWS ÜZERİNDE GO KURULUMU**

Windows'un 32 veya 64 bitlik sürümü için uygun Go kurulum dosyalarını; http://golang.org adresinden indirebilirsiniz. Kurulum dosyası .msi uzantılı olarak sunulmaktadır.



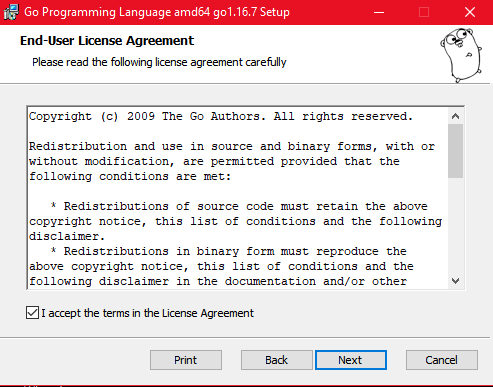
İndirdikten sonra, dosyanın üzerine çit tıklayarak kurulumu başlatabilirsiniz. Varsayılan kurulum ayarlarını değiştirmeden, Next, Next, ... şeklinde kurulum programını tamamlayabilirsiniz.

1. Adım: (Go’nun son sürümü)



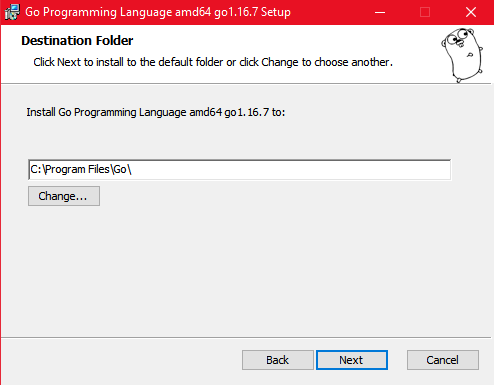
1. Adım:

(Lisans anlaşmasını kabul ediyoruz.)



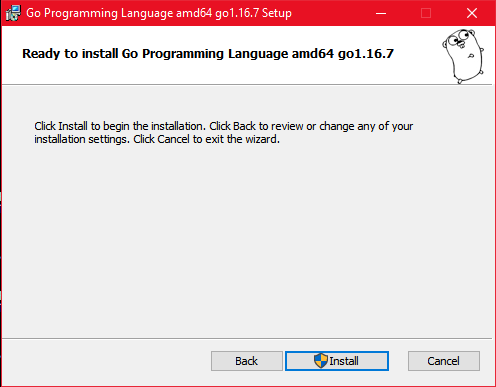
1. Adım:

(Programı yüklemek istediğimiz yer)



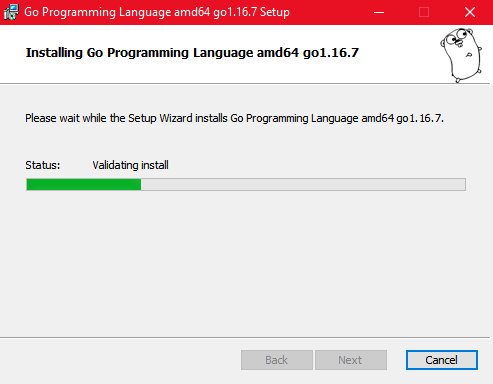
1. Adım:

(Yükleme sürecimizi başlatalım)



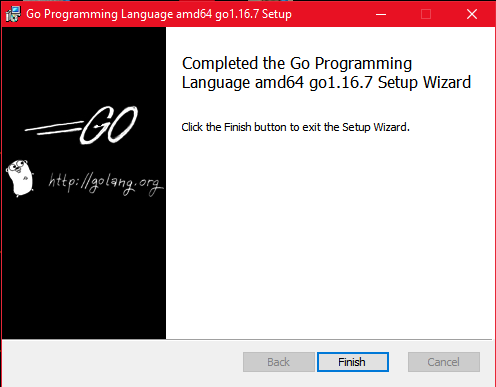
1. Adım:

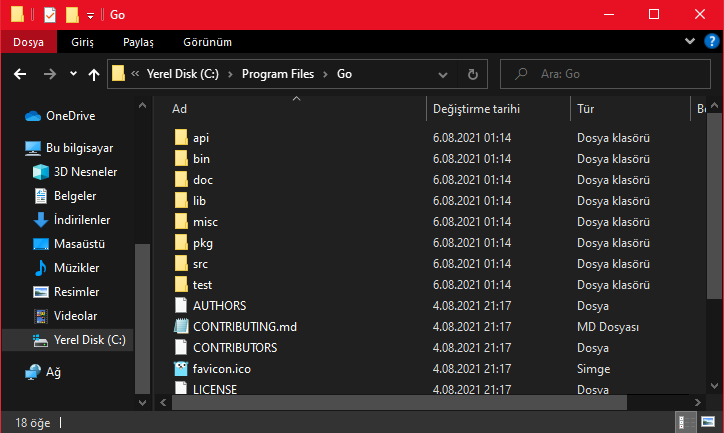
(Programın yüklenmesinin bitmesini bekliyoruz)



1. Adım:

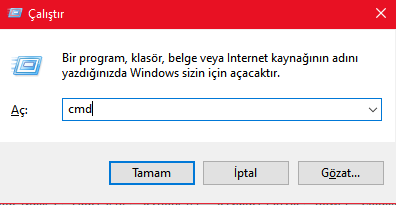
(Programın yüklenmesini tamamlıyoruz)





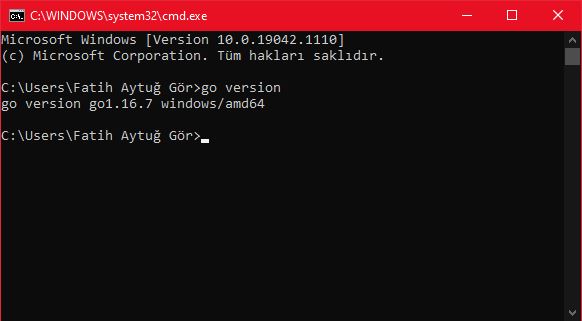
Varsayılan ayarlarla sihirbaz tamamlandığında, C:\Go\ klasöründe kurulmuş olacaktır. Şimdi Go’nun çalışıp çalışmadığını kontrol edelim.

1.Aşama



Windows tuşu + R tuşuna birlikte basarak Çalıştır penceresini açıp cmd yazıp Enter’a basıyoruz.

1. Aşama



Komut penceresi açıldığında “go version” yazıp Enter’a bastığımızda bize yüklediğimiz Go programının sürümünü göstermesi gerekiyor.

## **2.6.** IDE Araçları, Kurulumu, Kullanımı

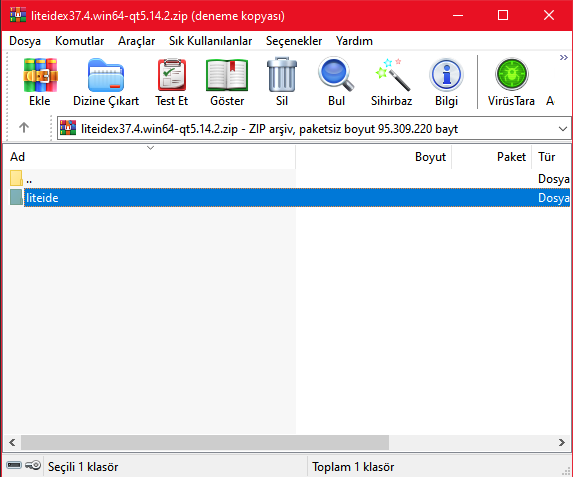
Eğer buraya kadar sorunsuz geldiysek, şimdi not defterinden kurtulup daha uygun bir editör kurabiliriz. Daha önce de bahsettiğimiz gibi, Go dili ile kullanılabilecek çok sayıda editör bulunmaktadır. Örneğin: Visual Code ve GoLand bunlardan bazılarıdır. Go ile kullanılabilecek editörlerin listesi, arama motorundan güncel olarak elde edilebilir. Bu editörlerden bazılarında Go kodu yazmak için eklenti kurulması gerekmektedir. Hangi editör en iyisidir? sorusunun net bir cevabı yoktur. Öyle olsaydı, sürekli yenileri geliştirilmezdi. Programlama amacıyla kullanılacak olan bir editörden beklentilerimiz; kod renklendirme, kod tamamlama, editör içinde derleme yapabilme, kodları adım adım çalıştırabilme (breakpoint), proje dosyalarını yönetme, hata bulma işlemleri gibi özelliklerdir. Bir başka programa da not defteri en güzel editördür diyebilir. Unix tarzı işletim sistemlerinde çokça kullanılan konsol editörü Vi ile kod yazan çok sayıda programcı vardır. Vi editöründe fare kullanılmaz; tüm işlemler klavye kısayollarla yapılır. LiteIDE tercih etmemizin iki önemli nedeni var:

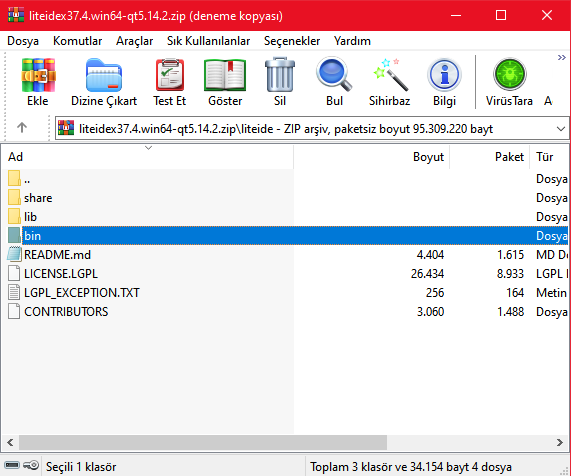
* Tüm popüler işletim sistemlerinde çalışabiliyor. Bu tarz uygulamalara cross-platform denmektedir.
* Doğrudan Go için yazılmış ilk editördür. Liteine kurulumu oldukça kolaydır. Önce LiteIDE'nin geliştirildiği [sourceforge.net](file:///C:\Users\Fatih%20Aytuğ%20Gör\Desktop\sourceforge.net) sitesinden son sürümü indirilmelidir.

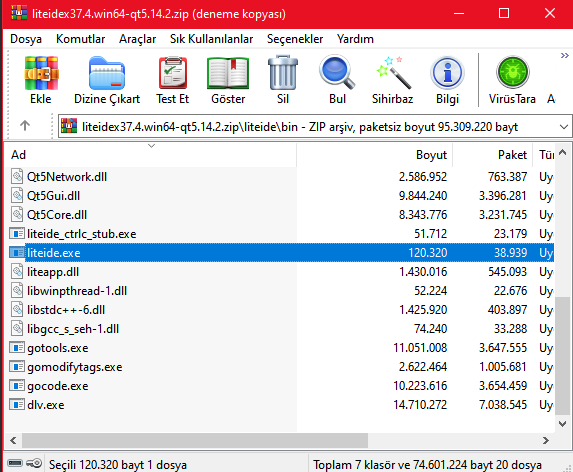


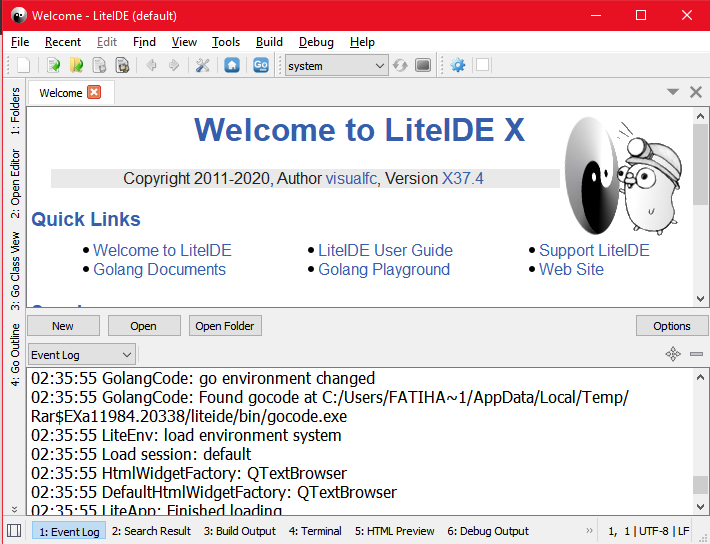
### 2.6.1 LITEIDE KURULUMU

LiteIDE kurulmayan bir geliştirme ortamı. İndirilen sıkıştırılmış dosya açıldıktan sonra, liteide isimli bir klasör çıkıyor. Bu klasörün içindeki bin klasörü içinde bulunan, liteide.exe dosyasına çift tıklayarak çalıştırabilirsiniz.



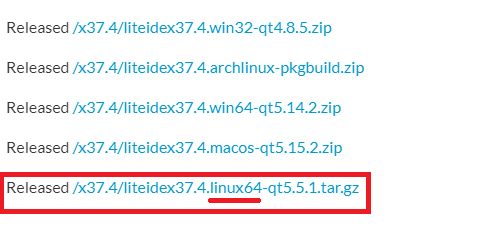




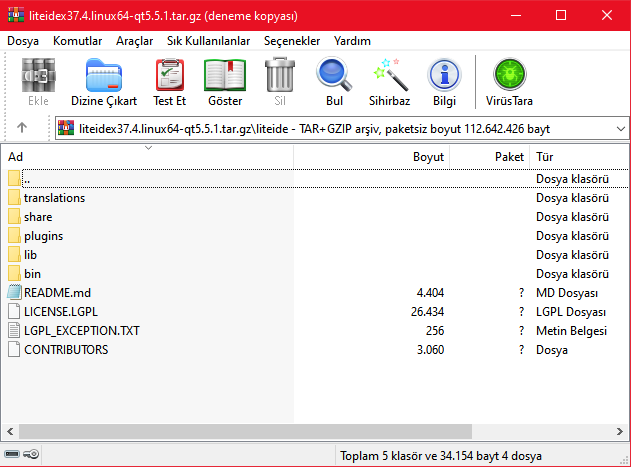


### 2.6.2. LINUX ÜZERİNDE LITEIDE KURULUMU

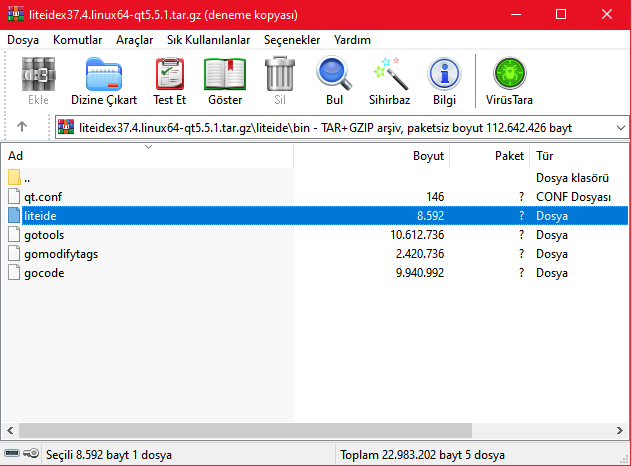
LiteIDE sourceforge üzerinde geliştirilmektedir. İşletim sistemimize uygun olan (32bit/64bit) sürümü doğrudan indirme sitesinden indirebiliriz.



LiteIDE sıkıştırılmış dosya biçiminde indirilip açıldıktan sonra şu şekilde bir klasör yapısı çıkmaktadır:



LiteIDE klasörü içinde bulunan bin/liteide isimli dosya, editörün çalıştırılabilir dosyasıdır. Doğrudan bu dosyaya çift tıklayarak çalıştırabilirsiniz.



### 2.6.3. APPLE MAC-OS ÜZERİNDE GO VE LITEIDE KURULUMU

Apple macOS (OS X) platformunda da Go kurulumu için doğrudan kurum yapilabileceği gibi, GVM kullanılarak ta kurulum yapılabilmektedir. macOS üzerinde de Ubuntu'daki apt komutuna benzer bir olanak kullanılabilmektedir. Programların kolay kurulması için kullanılan bu tarz araçlardan en yaygın olanı Homebrew isimli uygulamadır. Bu nedenle öncelikle Homebrew ve yazılım geliştirme araçlarını içeren XCode bileşenlerinin kurulumu yapılmalıdır. Öncelikle xcode-select --install komutu ile xcode yüklemesi yapılmalıdır.

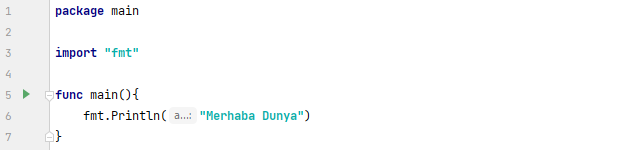
Xcode yükleme penceresi çıktığında Yükle düğmesine basılarak yükleme başlatılabilir. Yükleme işlemi bitince bir pencere ile durum bildirilecektir.

Xcode sağlıklı bir şekilde yüklendikten sonra, Homebrew yüklemesine geçilebilir. Homebrew için kendi sitesinde, Apple'ın ihtiyaç duymadığı ama size lazım olabilecek paketleri kurmanızı sağlar şeklinde açıklama yapılmıştır. <http://brew.sh> şeklindeki web sitesinde detaylı bilgi bulunabilir. Kurulumu kolay bir şekilde yapmak için, konsolda şu komutun yazılması gereklidir.

ruby -e "$(curl -fSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)"

## HelloWorld Uygulaması ve Adım Adım Çalıştırılması

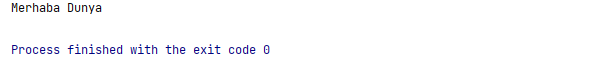
Go yazım bicimidline biraz benzemektedir. Kod blokları küme parantezleri içinde yazılır. C'den farklı olarak, satır sonlarında noktali virgül karakteri bulunmaz. Aşağıda örnek bir Go program kodu verilmiştir:



Yukarıda verilmiş olan program derlenip çalıştırıldığında, ekranda Merhaba Dünya! yazacaktır. Go'nun kodlama yapısından bahsetmek için verilen bu kısa program kodunu biraz inceleyelim:

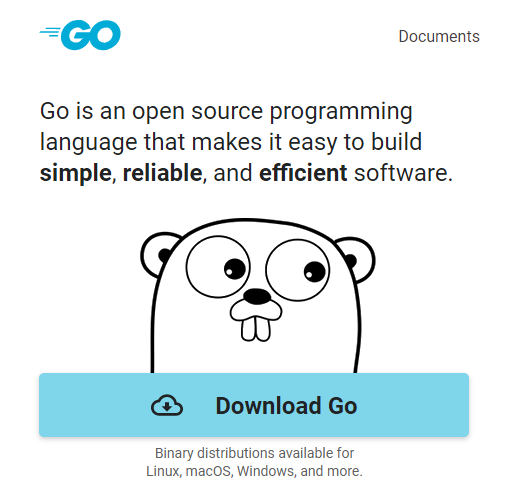
Gerektiğinde çağrılan, istendiğinde yeniden kullanılabilen kod bloklanna Go dilinde package (paket) ismi veriliyor. Import (dahil et) ifadesi de bir paketi, yazdığımız programa dahil etmek için kullanılıyor. Örnekte kullanılan fmt paketi, biçimlendirme ve ekrana yazdırma için kullanılmaktadır. Func sözcüğü, fonksiyon tanımlamak için kullanılmaktadır. Burda func main() ifadesi ile ana program bloğu tanımlanmış oluyor.

Bir Go Programı çalıştırılmak istendiğinde, çalıştırılmaya ilk başlanan yer, main paketi içerisindeki main fonksiyonu olacaktır. Programı derleyip çalıştırdığımız zaman programın ekran çıktısı şu şekilde olacaktır.



# ****3. Go Programlama Dili****

Go'nun resmî web sitesinde ( <https://golang.org/> ) Go şöyle tanımlanmıştır: "Go; basit, kararlı ve verimli yazılım oluşturmayı kolay bir şekilde sağlayan açık kaynaklı bir programlama dilidir". Go için sıklıkla "pragmatik (faydacı, işlevsel)" ifadesi kullanılmaktadır. Bunun sebebi gerçek dünya problemleri için tasarlanmış olmasıdır. Bilimsel çalışmalar veya farklı özel amaçlar Go'nun öncelikli odağı değildir. Go sadelik ve faydayı, programcılık geleneklerinden önde görür. İçerisinde paralel (çok görevli) çalışma özellikleri barındıran, performans ve basitlik odaklı, genel amaçlı bir dildir. Çok farklı alanlarda Go ile uygulama geliştirmek mümkündür. (Murat Özalp, 2017)



Projenin 3. Kısmını oluşturan Go Programlama Dili bölümünde IDE olarak Jetbrain’in Goland Programı kullanıllmıştır. Go’yu kullanmak için bilgisayarınıza herhangi bir IDE kullanmadan Golang’ın resmi websitesinde yer alan Online Compiler kullanılabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-1-1-a: <https://golang.org/> sitesinde yer alan Online Compiler

## ****3.1. Kullanıcı Etkileşimi****

Diğer dillerde olduğu gibi Go Programlama Dilinde de kullanıcıdan giriş alıp çıktı verme işlemleri vardır. Bu şekilde Go Programlama Dili interaktif bir hale dönüşür. Golang’da output ve input işlemleri Fmt Paketi’nin içerisinde tanımlanmıştır. Bu fonksiyonların kullanılabilmesi için bu paketin import edilmesi gerekir.

### 3.1.1. Output İşlemleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-1-1-a: Print ve Println fonskyionuyla Output Verme



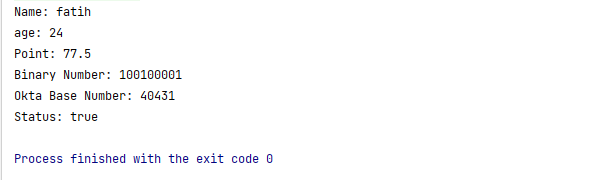
Örnek 3-1-1-a Ekran Çıktısı

Go Programlama dilinde output verme işlemi print ve println fonskyionuyla ilkel bir şekilde yapılabilir. Bu fonksiyonlar varsayılan format düzenini kullanır. Println fonksiyonunun print fonksiyonundan farkı yazdırma işlemi bittikten sonra bir alt satıra inmesidir.

Golang’da output verme işlemleri için en sık kullanılan fonksiyon printf fonksiyonudur. Bu fonksiyon sayesinde istenilen format düzeninde değişkenler yazdırılabilir. Ayrıca println fonksiyonundan farklı olarak alt satıra inmez. Alt satıra inilmesi için metin sonuna “\n” ifadesi yazılır.



Örnek 3-1-2-b: Printf fonksiyonuyla formatlı çıktı verme



Örnek 3-1-1-b Ekran Çıktısı

Printf Fonksiyonun’da en sık kullanılan format biçimleri şu şekildedir.

* + %v – Değeri varsayılan
  + %d – Ondalık tamsayı
  + %g – Float sayı
  + %b – 2’li tabanda
  + %o – 8’li tabanda
  + %t – True-False
  + %s – String formatında belirtmek için kullanılır.

### 3.1.2. İnput İşlemleri

Go Programlama Dili’nde kullanıcıdan giriş almanın birden fazla yolu vardır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-1-2: Scan methodu kullanarak kullanıcıdan giriş alma



Örnek 3-1-2 Ekran Çıktısı

Scan methodu boşluğa kadar olan kelimeyi kaydeder. Örnek 3-1-1’de Scan methodu kullanılarak kullanıcıdan giriş alınması amaçlanmıştır String türünde “isim” değişkeni ve “yas” değişkeni tanımlanmış olup scan methoduyla klavyeye girilen terim isim değişkenine atanmıştır. Aynı şekilde scan metodulya kullanıcıdan veri alınıp yas değiskenine atama yapılmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-1-2-b: Scanf methodu kullanarak kullanıcıdan giriş alma



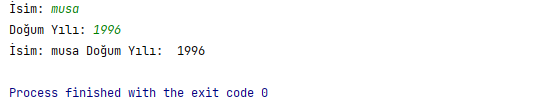
Örnek 3-1-2-b Ekran Çıktısı

Scanf Fonksiyonu da Printf Fonksiyonu gibi format içerir. Örnek 3-1-2-b’de string türünde isim ve yaş değişkenine scanf fonksiyonuyla iki değişken atanmıştır. İki atama arasına atılan boşluk atama işleminin sorunsuz gerçekleştirilmesini sağlar.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-1-2-c: Bufio Scanner kullanarak kullanıcıdan giriş alma



Örnek 3-1-2-c Ekran Çıktısı

Bufio paketi hafızaya alınmış giriş çıkış işlemlerini yapar. Ayrıca bu fonksiyon ile bir satır yazı input edilebilir.

## ****3.2. Koşullu Yapılar****

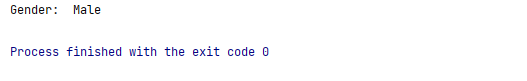
Çoğu programlama dili gibi Go Programlama Dili’de syntax özelliklerini C Programlama Dili’nden almıştır. Go Programlama dilinde koşullu yapılar 4 maddeden oluşur.

### 3.2.1. İf Bloğu

İf bloğu belirtilen koşulun sağlanması durumunda tek bir kod bloğunu çalıştırmak için kullanılır.



Örnek 3-2-3: İf Bloğu



Örnek 3-2-4 Ekran Çıktısı

İf koşulunda x her zaman true döneceği için ekran çıktısı bu şekilde olacaktır.

### 3.2.2. İf…Else Bloğu

Bir önceki üst başlıkta belirtildiği gibi tek koşullu ifadeler için if bloğu yazmak yeterliydi. Fakat birden fazla şart sunuluyorsa if else bloğu kullanılır. İf koşulu sağlanmadığı takdirde derleyici else bloğunu true olarak değerlendirir

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-2-2: İf…Else Bloğu



Örnek 3-2-2 Ekran Çıktısı

### 3.2.3. İf...Else İf...Else Bloğu

İf…Else if…Else bloğu birden fazla şartın olması durumunda koşulu sağlayan kod parçacığının çalıştırılmasında kullanılır

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-2-3: İf...Else İf...Else Bloğu



Örnek 3-2-3 Ekran Çıktısı

Örnek 3.2.3’te gösterildiği üzere if bloğunun koşulu sağlamaması durumunda diğer else if blokları kontrol edilir. Else if blokları da şartı yerine getirmez ise else bloğu çalışır.

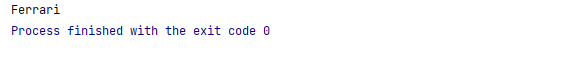
### 3.2.4. Switch-Case Bloğu

Go Programlama dilinde farklı koşullara dayalı olarak farklı eylemler gerçekleştirmek için Switch-Case ifadesini kullanırız. Bir değişkenin durumuna bağlı olarak daha çok fazla satır kaplayan kodlarda if-else bloğuna alternatif olarak kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-2-4: Switch-Case Bloğu



Örnek 3-2-4 Ekran Çıktısı

Switch Case bloğunda tüm koşulların sağlanmaması durumunda default koşulu çalışır.

## ****3.3. For Döngüsü****

**For Döngüsü** belirlitilen bir sayıdan başlayıp yine belirtilmiş bir sayıya verilen, koşul dahilinde ilerlemeye yarayan bir fonksiyondur. **For Döngüsünün** genel kullanım amaçları arasında dizi içindeki sıralı bilgileri bir değişkene yazmak veya ekrana bastırma gibi işlevleri de vardır. Kullanılış amacı olarak birden çok for döngüsü çeşidi vardır.

### ****3.3.1. Sınırlar Dahilinde Artan For Döngüsü****

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-5: Artan For Döngüsü

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-1: Ekran Çıktısı

Örnek 1’de, belirli sınırlar dahilinde artarak çalışan yaygın bir for döngüsü örneği görülmektedir. Diğer yaygın dillerin aksine koşul şartı metodun içine yazılmaz. Döngü parantezinin sol tarafına yazılır. For terimi yazıldıktan sonra koşul yazılır. Parantezin içerisine ise kod parçacığı yazılır. İnteger olarak tanıtılan bir “i” değişkenin 7’den küçük olması şartıyla kod parçacığını çalıştırması amaçlanır. Döngü çalışmasını bitirdiği zaman “i” değeri bir artar. Bu durumda for döngüsünün 7 kez çalıştığı gözlemlenir.

### ****3.3.2. Sınırlar Dahilinde Azalan For Döngüsü****

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-2: Azanlan For Döngüsü

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-2: Ekran Çıktısı

Örnek 1’de verilenin aksine koşul ifadesinde belirtilen değişken azalmak şartıyla çalışır. İstenilen indise ulaşılına kadar döngü çalışmaya devam eder. **3.3.3. While İşlevi Gören For Döngüsü**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-3: While İşlevi Gören For Döngüsü

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-3-3 Ekran çıktısı

Go programlama dilinde “while” terimi kullanılmamaktadır. Aynı işlevi gören For döngüsü formatı Örnek 3’te gösterilmiştir. İlk ataması “1” olarak verilen “toplam” değişkeni for döngüsünde 1000’den küçük olması şartıyla her defasında yeni değerle toplanmıştır.

### ****3.3.4. Sonsuz For Döngüsü****

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Örnek 3-3-4: Sonsuz For Döngüsü



Örnek 3-3-4 Ekran Çıktısı

Örnek 4’te koşul belirtilmedi için toplam değişkeni döngü içerisinde sonsuza kadar hesaplanır. Program sonsuza kadar çalışacağı için “Toplam” değişkeninin son hali ekrana yazılamayacaktır. Programın sağlıklı bir şekilde ekrana output vermesi için döngünün herhangi bir yerinde if bloğu ve break komutu eklemek faydalı olacaktır. Break komutu döngünün if bloğundaki koşulu sağlaması durumunda sona ermesini sağlar.

### ****3.3.5. Dizi Boyutu Kadar Çalışan Döngüler****



Örnek 3-3-5: Dizi Boyutu Kadar Çalışan For Döngüsü



Örnek 3-3-5: Ekran Çıktısı

Bir For Döngüsü’nü bir dizinin, Maps’in, Slice’nin ya da Channels’ın boyutu kadar döndürülmek istenmesi durumunda kullanılır. Anahtar kelimesi “Range”tir. Örnek 5’te verilen kod parçacığında “canlilar” diye dizi tanımlanmış olup bu diziye atamalar yapılmıştır. Dizinin indisi 0’dan başlayacak olup 5’e kadar devam edecektir. Bu durumda bu döngü 6 sefer çalışacak olup, ekran çıktısı yukardaki gibi olacaktır.

## 3.4. Listeler

### 3.4.1 Diziler

Dizi tek bir türdeki değişkenlerden oluşan ve aynı anda birden fazla değer tutabilen özel bir veri yapısıdır. Dizinin sahip olduğu değerlere element ya da item denir. Bir dizi belirli sayıda elemente sahiptir ve bu elementler Integer, String veya Boolean veya başka bir veri türünde olabilir. Bir dizinin indisi 0’dan başlar. Go programlama dilinde dizi oluşturma şekli şu şekildedir;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-4-1: Dizi oluşturma

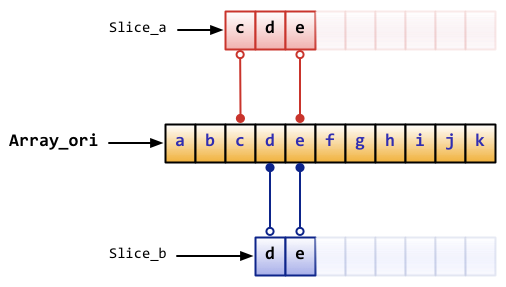
Örnek 3-4-1’de ilkel dizi oluşturma şekli verilmiştir. Var anahtar kelimesiyle birlikte dizinin ismi yazılır. Kapalı parantezin içine dizinin boyutu ve daha sonra bu dizinin veri tipi eklenir. Dizilere atama işlemi için dizinin indisine ihtiyac duyulur ve gerekli atama işlemleri yapılır.



Örnek 3-4-1: Ekran çıktısı

### 3.4.2. Slice

Dizi veri tipinin en büyük dezavantajlarından birisi boyutunun baştan belirtilmek zorunda olunması ve sonrasında değişiklik yapılamamasıdır. Bu kurallar Array Veri Tipi’nin ilkel olmasını, baştan planlanması gerektirilen ve tek türde veri türü barındırmasını şart koşan bir yapı haline getirmiştir. Eğer programın başında dizinin boyutunu bilinmiyorsa dinamiklik konusunda sıkıntılar yaşanabilir. Bu karmaşıklığın önüne geçmek için Slice Veri Tipi’ni kullanırız. Slice Veri Tipi dizilere nispeten daha dinamiktir ve bu sayede dizinin sonuna istediğimiz kadar elemen ekleyebiliriz.



Örnek 3-4-2-a: Slice mantığı

Go Programalama Dili’nde iki farklı şekilde Slice oluşumu vardır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-4-2-b: Make metodu ile Slice Oluşturma



Örnek 3-4-2-b: Ekran Çıktısı

Make fonksiyonunda kullandığımız ilk parametre veri türünü, ikinci parametre dizinin boyutunu ve üçüncü paramtre ise dizinin kapasitesini belirtir. Eğer kapasite belirtilmezse Go, kapasiteyi dizinin mevcut eleman sayısı olarak kabul edecektir.

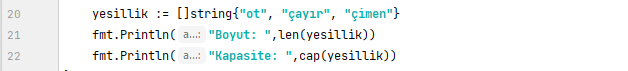
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-4-2-c: Başka Bir Yöntemle Slice Oluşturma

Örnek 3-4-2-b ve Örnek 3-4-2-c’de görüldüğü üzere dizimize eleman eklemek istediğimiz zaman Append Metodu’nu kullanırız. Append fonksiyonu ile dizinin sonuna eleman eklenir. Eğere yeni eklenen elaman dizinin mevcut kapasitesini aşıyorsa kapasite artırılıp sonra elemen ekleme işlemi gerçekleştirilir.

Bir Slice’nin boyutunu hesaplamak için len() fonksiyonu, kapasitesini hesaplamak için ise cap() fonksiyonu kullanılır. Boyut dizinin içindeki değerlerin sayısıdır. Kapasite ise dizinin maksimum alabileceği değer sayısıdır. Boş bir dilimin varsayılan (sıfır) değeri nil’dir



Örnek 3-4-2-d: Dizinin Boyutunu ve Kapasitesini Öğrenme



Örnek 3-4-2-d: Ekran Çıktısı

### 3.4.3. Maps

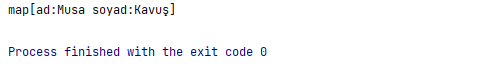
Go Programla Dili’nde verileri depolamak için birden fazla yöntem bulunabilir ama Maps’lerin çalışma mantığı diğerlerinden farklıdır. Maps’ler key ve value değerlerini aynı anda barındırmasıyla deplandıkları her alana bir bir anahter refere ederler.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-4-3-a: Maps Oluşumu ve Kullanımı

Oluşturulacak haritanın veri türü, parantez içindeki anahtar türü ve en son olarak değerin türü yazılır. Dizi ve Slicelarda olduğu gibi haritalarada parantez içine indis değeri yazılarak ulaşılabilir. Haritalar çalıştırılmadan önce bellekte oluşturulması gerekir. Bu yüzden Make Referans Ek’i kullanılır.



Örnek 3-4-3-a: Ekran Çıktısı



Örnek 3-4-3-b: Haritalarda delete fonksiyonu

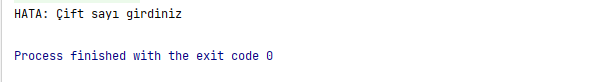
## 3.5. Hata Yakalama

Exception, türkçe anlamı istisnalar olan, istemeden de olsa kodlamada hata yapılan ve fark edilmeyen noktalarda istisnayi durumlarda nasıl bir müdahale edilmesi gerektiğini ele alan hatalar konusudur.  Programlarda bir hata ve istisna ile karşı karşıya kalındığında bu hatanın çözülmesi ve program akışının bozulmaması için bu kavramı kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

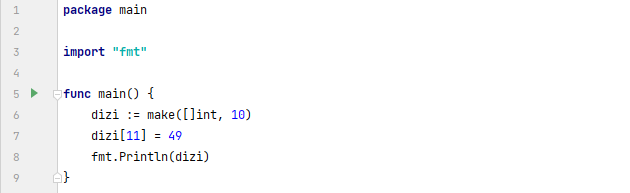
Örnek 3-5-1: Hata Yakalama Örneği



Örnek 3-5: Ekran Çıktısı

Örnek 3.5’te Hata Yakalama konusu ile ilgili örnek verilmiştir. Girilen sayının tek olup olmadığını kontrol eden, tek olmadığı durumunda hata mesajı veren program kodlanmıştır. Girdisi 4 olarak belirtilen değişken tekMi fonksiyonundaki if bloğundaki tek olma şartını sağlamadığı için program hata mesajı vermiştir.

Golang’da Error Handling için kullanılan anahtar kelimeler Panic ve Recover’dır. Panic hatasına bir örnek verilirse;



Örnek 3-5-2: Panic Hatası Örneği

Boyutu 10 olarak verilen diziye 11. Eleman olarak bir değer girilmesi istenmiş. Bu durumda taşma olayı olacağı için program Panic Hatası vermiş olacaktır. Bu programın çalıştırılması durumunda ekran çıktısı şu şekilde olacaktır;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-5-2: Ekran Çıktısı

Aşağıda panic oluşumana dair bir örnek daha verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-5-3: Panic Hatası

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-5-3: Ekran Çıktısı

Örnek3.5.3’te not adlı değişkene boş atama gerçekleştirildiği için program hata vermiş ve durmuştur.

## 3.6. Dosya İşlemleri

Dosya ve klasör işlemleri dendiğinde, bilgisayardaki depolama alanlarında (sabit diskler, USB harici bellekler, vb.) bulunan dosya ve klasörler ile ilgili işlemler kastedilmektedir. Bunlar; kopyalama, silme, taşıma, isim değiştirme, klasör oluşturma, dosya içine veri yazma, dosyadan veri okuma, vb. işlemlerdir.

Go’da dosya ve klasör işlemleri yapmak için os (Operating System – İşletim Sistemleri) ve io (input/output – giriş çıkış) paketleri kullanılmaktadır. Paketin detayları için <https://pkg.go.dev/os> ve <https://golang.org/pkg/io/> adreslerini ziyaret edebilirsiniz. Dosyalar ile yapılabilecek bazı işlemler aşağıdaki başlıklar altında toplanabilir.

### 3.6.1. Golang’de Dosya Oluşturma İşlemleri

Bu başlık altında Go Programlama Dili’nde dosya oluşturma işlemi işlenecektir. Dosya açma işlemleri için “Create” fonksiyonu kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-1: Dosya Oluşturma İşlemi Örneği

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-1: Dosya Dizini

Eşitliğin sol tarafında yazan dosya ifadesi açılacak dosyanın “dosya” isimli değişken tarafından temsil edileceğini belirtmektedir. Dosyanın oluşturulamaması durumunda bir hata mesajıyla karşılaşılacaktır. Eğer dosya başarılı bi şekilde oluşturulduysa hata değişkenine bir değer atanmadığı için nil olarak kalacaktır. Defer Fonksiyonu sayesinde dosyanın açılaması durumunda programın akışını duracağı sonucuna varırız.

### 3.6.2. Golang’de Dosya Açma

Go Programlama Dili’nde dosya açma ve okuma işlemleri için Open ve Openfile fonksiyonları kullanılır. Open metodu, dosyaları sadece ReadOnly modunda açar. OpenFile metodu ise dosya üzerinde birçok işlemin gerçekleştirilmesini sağlar.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-2: Dosya Açma ve Okuma İşlemi Örneği

Dosyaları seçerken dikkat edilmesi gereken önce dosya ismi, ardından işlem tipi ve en sondada dosya izni belirtilmelidir. 3 çeşit işlem tipi vardır.

O\_ RDONLY: Dosyayı sadece okuma için (read-only) açar.

O\_WRONLY: Dosyayı sadece yazma için (write-only) açar.

O\_RDWR: Dosyayı hem okuma hem de yazma için (read-write) açar.

Dosya seçme işlemleri için bu üc parametreden birinin seçilmesi zorunludur.

### 3.6.3. Golang’de İçeriği Okuma

metin içeren bir resim

Açıklama otomatik olarak oluşturulduGo’da dosya içeriği okuma işlemleri Os ve IO paketleri ile yapılabilir. Aşağıda verilen uygulamada, program ile aynı klasötde bulunan metin.txt isimli dosyanın içeriği ekrana yazdırılıyor.

Örnek 3-6-3-1: Dosya İçeriği Okuma Örneği

Program içerisinde BaytKesit isimli bir değişken tanımlanmıştır. Bu kesitin boyutu (eleman sayısı) DosyaBilgisi.Size() fonksiyonu ile belirlenmektedir. Daha sonra dosya.Read(BaytKesit) satırında dosya içeriği okunup BaytKesit değişkenine aktarılmaktadır. Read() fonksiyonu dosya içeriğini okumak ASCII kodları ile okur. BaytKesit değişkeni içeriği ekrana yazdırılırsa, ASCII kodları görünür. Bu nedenle, dosya metnini anlaşılır şekilde ekrana yazdırmak için, okunanMetin := string(BaytKesit) ifadesi ile metin türüne dönüştürme işlemi yapılır.

OkunanMetin isimli değişkende dosya içeriğindeki veri aktarılmış ve daha sonra ekrana yazdırılmıştır. Program çalıştırıldığında elde edilen ekran çıktısı şöyle olmaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-3-1: Ekran Çıktısı

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu Programdaki değişkenlerin türlerini görmek istenilirse reflect fonksiyonu kullanılabilir. Reflect paketi import edildikten sonra programın sonuna şu satırlar eklenir.

Örnek 3-6-3-2: Açılan Dosyanın Tür Bilgileri

Bu kod parçaçıkları kullanılan 3 değişkenin veri türünü ekrana yazdıracaktır. Program yeniden derlendiğinde ekran çıktısı aşağıdaki şekilde olacaktır.



Örnek 3-6-3-2: Ekran Çıktısı

Ekran çıktısında görülen []uint8 veri türü, yukarıdaki programın kodlarında değişken tanımlarken kullanılan []byte türü ile aynıdır. Go geliştiricileri, []uint8 veri türüne byte ismini vermişlerdir. ASCII kodlama sisteminde her karakter 1 bayt’ı temsil eder. Bu sayeden 8 bit bile ifade edilebilecek olan karakter sayısı 2⁸ =256 olmaktadır. Bu nedenle byte kullanılmak istenen yerde işaretsiz 8 bitlik tamsayı(uint8) kullanılabilir. İkisi de aynı karşılığa denk gelmektedir.

Önceki uygulamada, Os paketi içinde tanımlanmış olan File veri türüne ait olan Read() fonksiyonu kullanılmıştı. Bu sefer Io/ioutil (Io paketinin alt paketi olan ioutil) paketi ile aynı işlem yapılacaktır. Bu pakette gelen fonksiyonlar, dosyadan veri okuma işlemini oldukça kolaylaştırmaktıdır. Önceki uygulamada dosyadan veri okumak için; önce dosya açılmış, dosya boyutu alınmış ve buna göre byte’lar halinde okunmuştu. Bu sefer, tüm dosya içeriğini tek seferde bir değişkene atılması amaçlanmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu programın çıktısı Örnek 3-6-3-1’teki uygulamayla neredeyse aynı çıktıyı verecektir. Ancak görüldüğü üzere oldukça sade bir biçimde ve daha az kod parçacığı kullanarak aynı işlem yapıldı.

### 3.6.4. Golang’de Dosyaya Veri Yazma

Go Dili’nde dosyadan okuma yapabilidiğimiz gibi, dosyaya veri yazmak için de hazır Go fonksiyonları mevcuttur. Aşağıda bir metin dosyasına veri yazan uygulama verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-4: Dosyaya Veri Girme Örneği

Örnek 3.6.4’te ekrana bir şeyler yazılmayacaktır ama oluşturulan test.txt dosyasına aşağıda görülüğü üzere istenilen metin girilecektir. Dosyaya veri girilememesi durumunda ekrana hata mesajı verilecektir. Defer fonksiyonu ile de program duracaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-4: Veri Girilen Text Dosyası

### 3.6.5. Golang’da Klasör İşlemleri

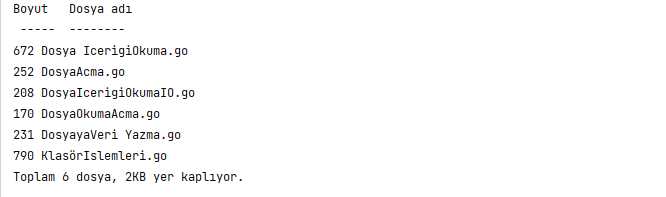
Dosya sistemindeki klasöre erişmek için, Os paketinden faydalılabilir. Aşağıdaki örnekte klasör içeriği listelemek için bir fonksiyon yazılmıştır. Listele ismindeki bu fonksiyona argüman olarak, listelenecek olan klasörün yolu verilmektedir. Fonksiyon dosya\_sayisi ve klasör\_boyutu isimli iki sayısal değişken döndürmektedir. Klasör içerisindeki dosyaların listesini almak için, dosyalar, \_ :=Klasör.Read-dir(-1) ifadesi kullanılmaktadır. Read-dir donksiyonu, listelenecek olan dosya sayısını argüman argüman olarak almaktadır. Bu şekilde dosyalar değişkenine aktarılan dosya listesi, range fonksiyonu ile işlenmekte ve her eleman için dosya.Name ve dosya.Size fonksiyonları ile isim ve boyutları ekrana yazdırılmaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-6-5: Klasör İşlemleri

Program çalıştırıldığında aşağıdaki gibi çıktı verecektir.



Örnek 3-6-5: Ekran Çıktısı

### 3.6.6. Golang’da Dosya Sorgulama

Go Programlama dilinde dizinde aratılan klasörün var olup olmadığı Os paketi ile sorgulanabilir.

metin içeren bir resim

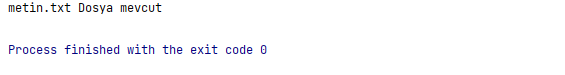
Açıklama otomatik olarak oluşturuldu

Örnek 3-6-6: Dosya Sorgulama

Dosya işlemleri için Os Paketi import edildikten sonra if-else akışında geçici değişken olarak kontrol değişkenine “metin.txt” stringi verilerek dosyanın ismi belirlendi. Eğer mevcutMu fonksiyonunda true değer dönerse “dosya mevcut”, false ise “Dosya mevcut Değil” mesajının alınması gerekir.



Örnek 3-6-6: Bilgisayar Konumundaki Dosya Dizini



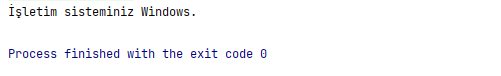
Örnek 3-6-6: Ekran Çıktısı

### 3.6.7. Golang’da Sürüm Kontrolü

Golang’da kullanılan sistemin sürümü kolaylıkla öğrenilebilir. Bunun için runtime.GOOS ile kontol yapılabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



Örnek 3-6-6: Ekran Çıktısı

## 3.7. Go ve Nesne Yönelimli Programlama

**Sınıflar** ve **Nesneler** nesne tabanlı programlamanın en önemli parçalıdır. Metod kullanımı, kalıtım gibi konular da nesne tabanlı programlama kapsamında kullanılan özelliklerdir. Biraz farklılık gösterse de diğer dillerde olduğu gibi Go dilinde de bu işlevler kullanılabilir. C, C++ ve Java gibi dillerde Classlar nesne tabanlı programlamanın temelidir. Go dilinde sınıflar yerine yapı (struct) kullanılmaktadır. Struct ve Sınıf kullanımı birbirine çok benzeyen yapılardır. Go dilinde nesne oluşturmak yerine yapı türünde değişken tanımlamak yeterli sayılır. Nesneye ait elemanlara ulaşmak için yine diğer nesne tabanlı dillerde olduğu gibi (.) işareti kullanılır. Go dilinde sınıf oluşturmak için yapı türü kullanılması zorunlu değildir. Farklı bir veri türü (tamsayı, metin gibi) kullanılarak da metotlar, kalıtım gibi özellikler kazanabilir. Yapı türünün kullanılma sebebi; altında çok sayıda alanda farklı veriler tutulma ihtiyacıdır.

### 3.7.1. Struct Kullanımı

Struct programlamada kaba tabirle birden fazla değişkenin bir arada paketlenmiş hali demektir. Farklı türdeki elemanları tek bir türde gruplandırmaya ya da birleştirmeye izin veren programlama yöntemidir. Örnek verilmesi gerekirse bir öğrencinin adı, soyadı ve numarası olur. Bu Structlar’la şu şekilde ifade edilebilir;



Örnek 3-7-1-1: Struct Oluşturma

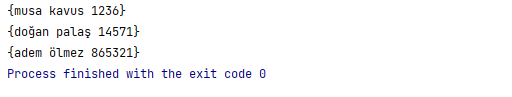
Type anahtar sözcüğü yeni bir tür tanıtır. Ogrenci adlı kümenin bir yapı olduğunun ifade etmek için struct kelimesi kullanılır. Struct küme parantezleri içindeki çeşitli alanların bir listesini oluşturur. Her alanın bir adı ve veri türü vardır. Yapının içindeki değişkenlere değer atamanın birden fazla yolu vardır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-1-2: Structlara Ulaşma ve Değer Atama

Birinci tanımlamada (satır10) değişken değerlerinin struct içerisinde oluşturulma sırasıyla yerleştirilmesi önem arzetmektedir. İkinci tanımlamada ise değişken isimleri verildiği için sıramala önem teşkil etmemektedir. Üçünü yöntemde ise ogrenci türünden ogr3 diye bir değişken tanımlanmış olup atama işlemleri ona göre yapılmıştır.



Örnek 3-7-1-2: Ekran Çıktısı

Struct değişkenlerini kullanırken her bir değişkenin normal olarak bir ismi olması gerekmektedir. Bazen değişkene isim vermek yerine değişkeni dizi olarak tanımlamak ve ismi de yapının bir parçası olarak kullanmak daha kullanışlı olabilir. Aşağıdaki kod parçacığında bu kullanıma bir örnek verilmiştir;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-1-3: Struct Dizisi Oluşturma

Örnek 3-7-3 de memurlar veri türünden memur adında 3 boyurlu bir dizi tanımlanmış bunun 1 numaralı indisine atama işlemi gerçekleştirilmiştir.

### 3.7.2. Metot Kullanımı

Metotlar, Struct gibi özel değişken türlerine bağlı olarak kullanılabilen özel fonksiyonlardır. Metot ve fonksiyon ibaresi birbirlerine çok karıştırılıyor olsa da aralarındaki fark şudur. Metot, fonksiyondan farklı olarak içinde parametre içerir. Parametre sayesinde metot, alıcının özelliklerine erişebilir. Programda bir metot oluşturulduğu zaman alıcı ve alıcı tipi aynı pakette olmalıdır. Ayrıca int, string vb. tiplerde dahil olmak üzere alıcı tipinin zaten tanımlı olduğu bir metot oluşturulmasına izin verilmez. Bu yapılmıya çalışılırsa derleyici hata verecektir. Kabataslak olarak metot oluşumu şöyledir;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-2-1: Metot Oluşumuna Örnek

Metotlar birden fazla türde olabilir. **Struct tipinde alıcı metota** örnek verilmesi gerekirse;

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Örnek 3-7-2-2: Struct Tipinde Alıcı Metot

Ogrenciler adında bir Struct oluşturulmuş ve bu struct isim, soyisim ve numara değişkenlerini içermiştir. Daha sonra goster isimli bir metot oluşturulmuştur ve alıcısı ogrenciler yapısından alınmıştır. Main fonksiyonu içerisinde Struct elemanlarına değerler atanmış ve goster adlı metot çağırılmıştır. Bu durumda ekran çıktısı şu şekilde olacaktır;



Örnek 3-7-2-2: Ekran Çıktısı

**Struct tipinden olmayan alıcı metot,** tür ve metot tanımları aynı pakette olduğu sürece, struct olmayan tür alıcı ile bir metot oluşuma izin verir. Eğer int, string gibi farklı paketlerde bulunurlarsa, farklı paketlerde tanımlandığı için derleyici hata verecektir. Örnek vermek gerekirse;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-2-3: Struct Tipinde Olmayan Metot Oluşumu

“sayi” isminde integer bir tip tanımlanmış olup main fonksiyonu içerisinde sayi1 ve sayi2’ye onun cinsinden değerler atanmıştır. Atanan bu değerler goster adlı metotta parametre plarak atanmış olup bu metot sonrasında tekrardan main fonksiyonu içerisinde çağırılmıştır.



Örnek 3-7-2-3: Ekran Çıktısı

**Pointer tipinde alıcı metot,** bir işaretçi alıcısı olan bir metot oluşumuna izin verir. İşaretçi alıcısı olan metotlar, alıcının işaret ettiği değeri değiştirebilir. Metotların genellikle alıcılarını değiştirmesi gerektiğinden, işaretçi alıcıları değer alıcılarından daha sık kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-2-4: Pointer Alıcılı Metod

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-2-4: Ekran Çıktısı

### 3.7.3. Fonksiyon Kullanımı

Fonksiyonlar genellikle, kullanıcıya aynı kodu yeniden kullanma imkânı veren, aşırı bellek kullanımından tasarruf sağlayan, zaman kazandıran ve en önemlisi kodun daha iyi okunabilirliğini sağlayan kodlar ve ifadeler bloğudur. Genel olarak fonksiyon, belirli bir görevi yerine getiren ve sonucu kullanıcıya döndüren kod parçacıklarıdır. Bir fonksiyon kullanıcıya değer döndürmeden de belirli bir görevi gerçekleştirebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-1: Fonksiyon Kullanım Taslağı

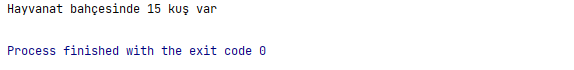
Örnek 3-7-3-1’de fonksiyonların kullanım taslağı verilmiştir. Func ifadesi, fonksiyon oluşturmak için genel ifadedir. Kullanılacak fonksiyonun ismi bildilirilir. Parametre listesi, parametrelerin ismini ve türünü gösterir. Return tipi, kullanıcının insiyatifine bırakılmıştır. Fonksiyonda return tipi kullanılıyorsa, fonksiyonun içinde return ifadesi kullanılması zorunludur. Oluşturulan fonksiyonun çağrılması gerekir. Buna bir örnek verilirse;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-2: Basit Fonksiyon Çağırımı

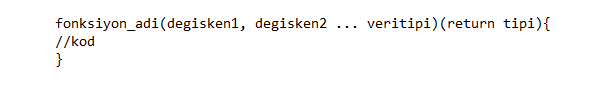
Fonksiyon çağrısı, kullanıcı fonksiyonu çağırmak istediğinde kullanılır. Örnek 3-7-3-2’de gösterildiği gibi iki parametreli hayvanatBahçesi isimli fonksiyona kuş ve adet isimli iki değişken gelip string türünde bir return ifadesi döndürmüştür. Daha sonra main fonksiyonunda, hayvanatBahçesi adlı fonksiyon çağrılıp dönüş değeri string türünde sonuc değişkenine aktarılmıştır. Bu durumda ekran çıktısı şu şekilde olacaktır;



Örnek 3-7-3-2: Ekran Çıktısı

Fonksiyonlar kendi içlerinde Variadic fonksiyonlar, Anonim fonksiyonlar, Main ve init fonksiyonlar olmak üzere dörde ayrılır.

**Variadic Fonksiyonlar,** bilinmeyen sayıda parametre geçişine izin veren fonksiyon çeşidir. Son parametrenin türünden önce üç nokta (…) gelir. Bu Fonksiyonun bu tipteki herhangi bir sayıda parametreyle çağrılabileceğini gösterir. Taslak biçimi şöyledir;



Örnek 3-7-3-3: Variadic Fonksiyon Kabataslak Gösterimi

Fonksiyonun içinde …type bir slice vardır. Variadic bir fonksiyonda mevcut bir slice’da iletilebilir. Variadic fonksiyonda herhangi bir parametre iletilmediğinde fonksiyonun içindeki slice’nin için boşmuş gibi davranılır. Bu fonksiyon türünde birden çok Slice’da iletilebilir. Variadic parametre dönüş değeri olarak kullanılamaz ama Slice olarak döndürülebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-4: Variadic Fonksiyon Örneği

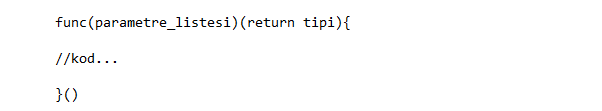
Join ifadesiyle variadic fonksiyona stringler eklenir. Örnekten anlaşılağı üzere parametre sayısı bildirmeden fonksiyona istenildiği kadar veri aktarılabildi. Bu durumda Örnek 3-7-3-4’ün ekran çıktısı şu şekilde olacaktır.



Örnek 3-7-3-4: Ekran Çıktısı

Bilinmesi gerekir ki Variadic Fonksiyon sadece, parametre sayısı bilinmeyen kod parçacıklarında kullanılır. Programlarda bu fonksiyon türü kullanıldığında programın okunabilirliği artar.

**Anonim fonksiyon** herhangi bir ad içermeyen fonksiyondur. Yazıldıkları yerde hemen çalışırlar. Diğer fonksiyonlar gibi parametre içermedikleri için fonksiyonun sonunda parametre verilerek çalıştırılırlar.



Örnek 3-7-3-5: Anonim Fonksiyon Taslağı

Anonim fonksiyonlar parametre veya return değeri almayabilir. Bulundukları yerde çalıştıkları için fonksiyon içinde fonksiyon olarak çalışırlar.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-5: Parametresiz Anonim Fonksiyon Örneği

Parametre alan fonksiyona örnek verilirse;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-6: Parametreli Anonim Fonksiyon Örneği



Örnek 3-7-3-6: Ekran Çıktısı

**Main Fonksiyon,** çalıştırılabilir programlarla birlikte kullanılan özel bir pakettir. Tüm çalıştırılabilir programların giriş noktasıdır. Herhangi bir parametre almaz ve hiçbir değer döndürmez. Program çalıştırıldığında Go derleyicisi otomatik olarak Main Fonksiyonunu çalıştırır. Bu yüzden Main Fonksiyonunu çağırmaya gerek yoktur ve her program bu fonksiyonu içermelidir.

**İnit Fonksiyonu** bir bakıma Main Fonksiyonu gibidir.Herhangi bir parametrealmaz ve hiçbir şey döndürmez. Bu fonksiyon her pakette bulunur ve paket başlatıldığında bu fonksiyon çağırılır. Bu fonksiyon örtük bir fonksiyondur, bu nedenle ona herhangi bir yerden ulaşamazsınız. Aynı programda birden fazla İnit fonksiyon olabilir ve bunlar oluşturuldukları sıra ile yürütülür. İnit Fonksiyonu, main fonksiyonunda önce çağrılır. Bu yüzden main fonksiyonun çalıştırılmasına bağlı değildir. Bu fonksiyonun amacı, global bağlamda başlatılamayan global değişkenleri başlatmaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-3-7: İnit Fonksiyon Örneği

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 3.7.4. İnterface

Go Dili’nde arayüzler diğer dillerden farklıdır. Arayüz, bir veya daha fazla metod bilidirmeden kullanılan özel bir türdür ve soyuttur. Soyut olduğu için nesnesinin oluşturulmasına izin verilmez. Ancak arayüz türünde bir değişken oluşturmanıza izin verilir ve bu değişkene, arayüzün gerektirdiği metodlara sahip somut bir tür değer atanabilir. Kısaca arayüz, özel bir tür olduğu kadar aynı zamanda bir metodlar topluluğudur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-4-1: İnterface Taslak Biçimi

Type kelimesi kullanıldıktan sonra oluşturulacak İnterface’nin ismi yazılır ve bunun bir arayüz olduğunu bildirmek için “interface” terimi yazılır. Son olarak süslü parantezlerin içine metodların imzaları yazılır.



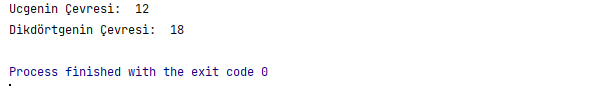
Örnek 3-7-4-2: İnterface Basit Kullanım Tipi

Golang’da, bir arayüzün uygulanması için arayüzde bildirilen tüm yöntemlerin uygulanması gerekir. Go Dili arayüzleri dolaylı olarak uygulanır ve diğr diller gibi arayüz uygulamak için belirli bir anahtar kelime içermez.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-4-3: İnterface Kullanımına Örnek

Örnek 3-7-4-3’ te görüldüğü üzere veri tipinde ucgenCevre ve dikdortgenCevre diye iki metod tanımlanmış olup cevre formülüne değerler atanmıştır. Main fonksiyonu içerisinde data yapısının içine değişkenler verildikten sonra metodlar çağırılmıştır.

Örnek 3-7-4-3: İnterface Ekran Çıktısı

Arayüzlerle ilgili bilinmesi gereken diğer şeyler, arayüz hem statik hemde dinamik yapıda olabilir. Statik tip, arayüzün kendisidir. Ancak arayüzün statik bir değeri yoktur, bu yüzden her zaman dinamik değere işaret eder.

### 3.7.5. Eş Zamanlılık

Go Dili’nin büyük sorunlara kolay çözüm getirmek gibi bir amacı vardır. Eş zamanlı görev çalıştırmanın çok kolay olması da Go’nun bu amaçla sunduğu güzelliklerden bir tanesidir. Güncel dillerin birçoğunda çoklü göre çalıştırma olanağı vardır. Ancak Go, bu konuda kendine has, güzel bir yöntemi vardır. Eş zamanlı görevleri **Gorutin** (Goroutine) ismi verilen parçalar halinde işletir. Gorutin kullanımında bellek ihtiyacı daha küçüktür ve süreçlerin yöntemi daha kolaydır. Go Dili’nde bu kolaylıklardan faydalanmak için eş zamanlı olarak çalışması istenen fonksiyonlarda, fonksiyon çağrılırken başına “go” ifadesi eklenmesi yeterlidir.



Örnek 3-7-5-1: Gorutin Fonksiyonu Taslağı

Gorutin kullanımı birden fazla yerden veri beklendiği zaman çok işe yaramaktadır. Örneğin birkaç web sitesinden veri çekmek gerektiğinde, farklı veritabanı sunucularından veriler alınması gerektiğinde veya birden fazla sensörden aynı anda veri çekilmek istendiğinde, işlerin birbirini beklememesi için kullanılabilir.

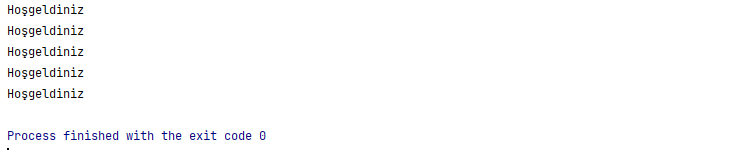
Aşağıda Gorutin kullanımına ait bir örnek gösterilmiştir. Goster isimli bir fonksiyon String cinsinden bir x değeri almıştır. Daha sonra main fonksiyonu içerisinde bu fonksiyon gorutin biçiminde ve normal biçiminde çağrılmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-5-2: Gorutin Kullanımına Örnek

Bu programın ekran çıktısı şu şekilde olacaktır.



Örnek 3-7-5-2: Ekran Çıktısı

Bu örnekte “goster” isimli fonksiyon iki farklı şekilde çağrılmıştır. İlki Gorutin şeklinde çağrılmış ve paramtre olarak “Merhaba” metni girilmiştir. Diğer çağırım şeklinde ise parametre olarak “Hoşgeldiniz” metni verilmiştir. Fakat burda görüldüğü üzere sadece normal fonksiyon çağırımının sonucu ekrana gösterilmiştir. Gorutin fonksiyon çağırımı ekrana çıktı olarak verilmemiştir. Çünkü bir Gorutin fonksiyonu çağırıldığında bu çağrı hemen geri döner. Burda Gorutin fonksiyonun çalışmasını bitirmesi beklenilmez. Fonksiyon normal çağırımda daha erken görevini bitirdiği için program sonlanmıştır. Bu programın normal şekilde çalışmasını sağlamak için;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-5-3: Gorutin Kullanımını Düzünlenmiş Hali

Burda eklenen time.Sleep () fonksiyonu 1 saniyelik süre zarfı farkıyla her iki fonksiyonunda çağırılmasını sağlar. Bu işlem For Döngüsü içerisindeki şart sağlanana kadar yani i, 5’ten küçük oluncaya dek devam eder. Dikkat edilmesi gereken burda hem Gorutin hem de normal çağrıım şekli aynı anda sırayla çalışır. Yeni durumda ekran çıktısı şu şekilde olacaktır;



Örnek 3-7-5-3: Ekran Çıktısı

Gorutin kullanmanın avantajları arasında;

Gorutin yığınlarda depolanır ve yığının boyutu programın ihtiyacına göre büyüyüp küçülebilir. Ancak iş parçacıklarında yığının boyutu sabittir. Gorutin’de herhangi biri, kaynak gereksinimi nedeniyle iş parçacığını engellerse, kalan tüm Gorutinler, yeni oluşturulan bir işletim sistemi iş parçacığına atanır.

Gorutin kullanımı anonim şekilde de çalıştırılabilir. Buna örnek verilirse;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

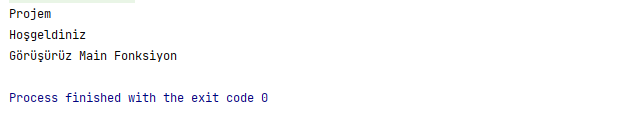
Örnek 3-7-5-4: Anonim Gorutin Taslağı

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-5-5: Anonim Gorutin Kullanımı

Go Dili’nde anonim bir fonksiyon için Gorutin kullanılabilir. Tek yapılması gereken başına “go” kelimesinin eklenmesidir. Bu şekilde Gorutin anonim fonksiyonda çalıştırılmış olur. Bu durumda ekran çıktısı böyle olacaktır;



Örnek 3-7-5-5: Ekran Çıktısı

### 3.7.6. Kanallar

Eş zamanlı çalışan fonksiyonların birbiriyle haberleşmeleri için **kanal** ismi verilen yöntem kullanılır. Haberleşme başlamadan önce iki fonksiyon aralarında haberleşme kanalı oluşturulmalıdır. Daha önceki bölümlerde ele alınmış olan **Maps** ve **Slice** kullanımında olduğu gibi burada da Make komutu ile kanal oluşturulur. Kanal oluşturmak için “chan” ifadesi kullanılır. Temel kullanım şekli aşağıda gösterildiği gibidir:



Örnek 3-7-6-1: Kanal Oluşturma Taslağı

Fonksiyonlar arasında taşınması gereken mesajın türüne göre kanal oluşturulmalıdır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-2: Go’da Kanal Kullanımı

Bu programda önce kanal1 := make (chan string) ifadesi ile “kanal1” isminde yeni bir kanal oluşturuldu. Kullanıclacak kanal metinsel ifadeleri String cinsinden taşıyacaktır. Kanalın nasıl tanımlanacağı örnek 3-7-6-1 nolu örnekte gösterilmişti. Burda go func () {kanal1 <­ “merhaba”} () satırında, kanal1 kanalına “merhaba” metni yerleştiriliyor. Veri aktarma işlemi için sola ok işareti (<-) kullanılmıştır. Bu satırda anonim (isimsiz) fonskiyon oluşturulmuştur. Kanal üzerinden haberleşme yapabilme kabiliyeti sadece Gorutinlerde vardır. Ancak Gorutini ayrı bir fonksiyon olarak çalıştırmak yerine, anonim olarak kod içinde kullanılması tercih edilmiştir. Bu fonksiyon sadece bir kez kullanılacağı için normal fonksiyon tanımlama şeklinde kullanılmamştır. Daha sonra kanal1’deki veri, mesaj değişkenine atanmıştır. Veri aktarma işlemi için yine (<-) işareti kulanılmıştır. Program çalıştırıldığında ekranda “merhaba” yazısıyla karşılaşılacaktır.

Close () fonksiyonu yardımyla bir kanal kapatılabilir. Bu yerleşik bir fonksiyondur. Bu yerleişk fonksiyona daha fazla değer gönderilemeyeceğini gösterir. Close () metodu içeren ve anonim olmayan bir kanal şöyle oluşturulur:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-3: Anonim Olmayan Fonksiyonda Close Fonksiyonu

Burda “fonksiyonum” isminde bir fonksiyon oluşturulmuş olup bu fonksiyona “Merhaba” metni atanmıştır. Daha sonra bu fonksiyon Gorutin metodula çağrılmış ve kanalın açık ya da kapalı olmasına göre ekrana çıktı verilmiştir. Ekran çıktısı şöyledir:



Örnek 3-7-6-3: Ekran Çıktısı

Bir kanalın varsayılan değeri nil’dir. Bir for döngüsü, kanal kapanana kadar kanala gönderilena ardışık değerler üzerinde yenilenebilir. Kanallarda gönderme alımı ya da gönderimi iptal edilebilir. Bir kanala veri gönderilidiğinde, diğer Gorutin o kanaldan okuyana kadar kontrol o gönderme ifadesine bloke edilir. Benzer şekilde, bir kanal Gorutin’den veri aldığında, “read” ifadesi bloğu başka bir Gorutin deyimine atar.



Örnek 3-7-6-4: Birden Fazla Kanal Kullanımı

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-4: Ekran Çıktısı

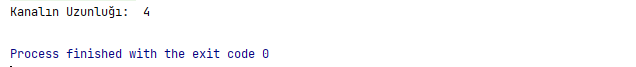
Bir kanalın uzunluğu Len Fonksiyonu kullanılarak öğrenilebilir. Burdaki zuunluk değeri, kanal arabelleğinde sıraya alınan değer sayısını gösterir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-5: Kanal Uzunluğunu Gösterme

Ekran çıktısı aşağıdaki gibidir.



Örnek 3-7-6-5: Ekran Çıktısı

Kanalın kapasitesi ise cap () fonksiyonu kullanılarak öğrenilebilir. Buradaki kapasite, arabelleğin boyutunu gösterir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-6: Kanalların Kapasitesini Bulma



Örnek 3-7-6-6: Ekran Çıktısı

Gösterildiği üzere bir kanal, aynı anda çalışan Gorutinler arasında birbirlerine veri gönderip alabilmeleri için bir iletişim aracıdır. Varsayılan olarak bir kanal çift yönlüdür ama tek yönlü kanal da oluşturulabilmesi mümkündür. Yalnızca veri alabilen ya da veri gönderebilen bir kanal tek yönlü kanal olarak adlandırılır. Tek yönlü kanal aşağıda gösterilidği make () fonksiyonu ile oluşturulabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-7: Tek Yönlü Kanal



Örnek 3-7-6-7: Ekran Çıktısı

Aynı şekilde çift yönlü bir kanal tek yönlü bir kanala da dönüştürülebilir veya başka bir deyişle, iki yönlü bir kanal yalnızca alıcı veya yalnızca gönderici bir kanala dönüşümü yapılabilir. Ancak bunun tersi mümkün değildir. Aşağıdaki programda gösterildiği gibi:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-7-6-8: Tek Kanala Çevirme



Örnek 3-7-6-8: Ekran Çıktısı

Unutulmamalıdır ki, tek yönlü kanal, programın tip güvenliğini sağlamak ve programın daha az hata vermesi için kullanılır veya yalnızca veri alabilen ya da gönderebilen bir kanal oluşturulması istendiğinde tek yönlü bir kanal oluşturulabilir.

## 3.8. Go ve Web Programlama

Web Programcılığı başlı başına bir çalışma alanıdır. 2000’li yılların başlarına kadar, Web Programcılığı dendiğinde, çoğunlukla ASP veya PHP dillerini kullanmak ve biraz da veritabanından anlamak diye yorumlanırdı. Oysa günümüzde standartları çok hızlı değişen, çok sayıda farklı bileşenler konuşulmaktadır. Web tabanlı sistemler çoğaldıkça, ihtiyaçlarda değişmekte ve programcılık sektöründe çalışanların da işi zorlaşmaktadır. Web kullanımı arttıkça; güvenlik, performans, geliştirme ve bakım kolaylığı gibi konuların önemi daha da artmaktadır. Güncel programlama dillerini geliştirenler, bu konularda daha çok kafa yormaktadır.

Önceki konularda Go Dili’nin güncel güncel ihtiyaçlara cevap vermek üzere tasarlandığından bahsetmiştik. Örneğin, eş zamanlı çalışma konusunda pratik şekilde çözüm üretmek üzere gerekli altyapı hazırlığı gelmekteydi. Aynı durum Web Programlama sistemi içinde geçerli olmuştur. Aşağıda basit yapıda bir web uygulaması gösterilmiştir:



Örnek 3-8-1: Go Dili’nde Basit Bir Web Programı

Web Programlama konusunda pratik çözümler sunan ve çokça kullanılan diğer diller gibi Go Dili de Framework (çatı) kullanarak harikalar yaratabilmektedir. Go Dili ile kullanılabilen Framework yapıları bulunabilmektedir. Beego, Martini, Gorilla ve Gin yaygın kullanılan Frameworklerden bazılarıdır. Aslında Go’nun yanında ayrı bir Web Framework’ü kullanmaya gerek yoktur. Çünkü Go ile gelen net/http ve html/template paketleri, Web Programcılığı için başlı başına yeterli ve güçlü araçlardır. Go ile Web Programcılığı yapmak isteyenlerin; dilin yapısına hâkim olabilmesi için ilave bir çatı kullanmadan doğrudan Go ile gelen paketlerle başlamaları tavsiye edilmektedir.

#### **3.8.1. Statik Web Sunucusu**

Statik web sayfaları, bir motor tarafından üretilmezler. Sayfa her çağırıldığında aynı cevap alınır. Herhangi bir yere veya değişkene bağlı değillerdir. Örneğin, stil dosyaları olan CSS dosyaları, resimler, videolar ve HTML dosyaları gibi dosyalar statik içeriklidir.



Örnek 3-8-1-1: Basit Bir Statik Web Sunucusu

Bu program çalıştırıldığında, TCP 8000 portu açılışında bir aksilik yaşanmazsa ekrana aşağıdaki gibi bir ekran çıktısı verilecektir;



Örnek 3-8-1-1: Ekran Çıktısı

Http.handle ile başlayan satırda, / şeklinde gelen klasör isteklerinin /statik web klasörüne yönlendirilmesi için işlem yapılmaktadır. Statik klasöründe index, html isimli, bu klasör içerisinde web üzerinden sunulacak olan dosyadır. Bu dosyanın web üzerinden çağırılması için, tarayıcıdan <http://localhost:8000/index.html> adresi kullanılmalıdır. Index.html ismi uzun zamandır web sunucularında varsayılan olarak kullanılmakta olduğundan, Go Dili’nde de eğer dosya ismi kısmı yazılmazsa yine aynı dosya getirilecektir.

#### **3.8.2. Html Şablon (Template) Kullanımı**

HTML şablonları, web sayfası görünümünü (tasarımını) belirten şablonlardır. Karmaşık işleri yapan program kodlarını (fonksiyonlarını) web tasarımındarı ayırmak için kullanılır. Bu sayede farklı uzmanlık alanları gerektiren işler birbi­rinden ayrılmış olur. Bir kişi sadece görsel tasarım ile uğraşırken, başka bir kişi de görsel tasarımda kullanılacak olan verilerin hazırlanması ve sunulması ile uğraşabilir.

Go dilinde HTML şablon kullanımı için html/template isimli bir paket gelmekte­dir. Daha önce kullandığımız net/http paketi ile, bu ikili oldukça güzel olanaklar sunmaktadır. Bu bölümde yapılacak örnek uygulamada iki tane dosya oluşturalacaktır. Bir tanesinde, klasik işleri yapan Go kodları yazılacaktır.

Bu uygulama özlü sözleri web sayfasına yazdırmaktadır. Go içerisinde yapı (stru-ct) biçiminde tanımlanmış olan özlü sözler, HTML şablon dosyasında uygun yerlere yazdırılıyor. Uygulamanın sade olması için yine birçok işlevsel özellik devre dışı bırakılmştır. Örneğin sadece bir tane özlü söz bulunmaktadır. Benzer şekilde, hiçbir hata denetimi yapılmamaktadır. Uygulamanın Go kodları aşağıdadır:

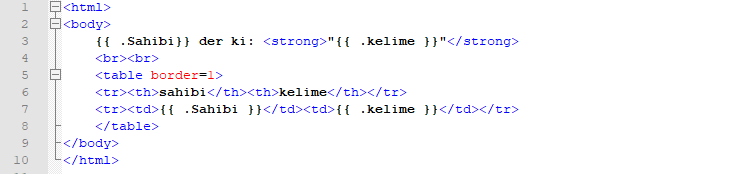
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-8-2-1: Go’da Şablon Kullanımı

Yukarıdaki Go programında, html/template paketi import edilmiştir. Özlü sözün hem söyleyeni hem de cümlesinin şablon dosyasına kolay bir şekilde gönderilebilmesi için, Söz isminde bir yapı (struct) oluşturulmuştur. Bu değişkenlere Go programı içerisinden ÖrnekSözl. Kelime ve sampleQuote.sahibi şeklinde erişilebilmektedir. log.Println satırında bu kullanım görülmektedir. Programda şablon dosyasının yüklenmesi ve okunması için şu satır yazılmıştır: şablon, \_ := template.ParseFiles("templates/index.html")

Üstteki satırda; templates klasörü altındaki index.html dosyası belleğe yüklenir, HTML kodlarının çözümlemesi yapılır ve HTML kodlarının bu durumu şablon isimli değişkene aktarılır. Aslında sablon, \_ := ... şeklinde kullanım yerine, şablon, hata şeklinde kullanılmalıdır. Ancak bu uygulamada hata denetimi yapılamadığı için hata olasılığını göz ardı edilmiştir. Sonraki satırda HTMLCevap ve OrnekSözl argümanları ile şablon.Execute fonksiyonu kullanılmıştır. Bu satır da belleğe yüklenmiş olan şablonun işletilmesi ve HTML cevabının oluşturulması için kullanılmıştır.



Örnek 3-8-2-1: İndex.html Dosyası

Yukarıdaki şablon dosyasını Go programı içinde işletirken, şablona argürnan olarak samplequote yapısını göndermiştik. Şablon dosyası içerisinde değişken isimlerinin yazılırken nokta (.) ile başladığına dikkat edelim. Şablona gönderilen samplequote yapısının içindeki alanlara, şablon içerisinden doğrudan {{.kelime}} şeklinde erişilebiliyor.

HTML şablon içerisinde Go ifadeleri kullanılmak istendiğinde bu ifadeler, ikili IPUCU küme parantezi içerisinde {{...}} şeklinde yazılır.

Örnek uygulama çalıştırılınca, Go konsolunda bir mesaj görünmüyor ancak Web sunucusu çalışmaya ve TCP 8000 portunu dinlemeye başlıyor. Tarayıcı üzerinden bu web sitesi açıldığında aşağıdaki ekran görüntüsü elde edilir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-8-2-1: Ekran Çıktısı

HTML şablon dosyasına gönderilen çoklu verilerde yine range fonksiyonunu kullanabiliriz. Bunun için şu tarzda bir yazım şekli bulunmaktadır:

{{range Sanahtar, $deger := . }}

{{$anahtar}} : {{$değer}}<br>

{{end}}

HTML şablon dosyasında range fonksiyonu ile döngü çalıştırılabilir. Bu sayede CU şablona gönderilen çoklu verilerin her birisi HTML şablonda ayrı ayrı işlenir.

Uygulamada kullandığımız şablona dizi şeklinde veri gönderip yukarıdaki yazım biçiminde döngüye sokarsak, web sitesinde şu şekilde çıktı vermektedir:

0: {Albert Einstein Her şeyi olabildiğince sade yapın, ama basit değil.}

1: {Albert Einstein Dehanın 10'da 1'i yetenek 10'da 9'u da çalışmaktır.}

Yukarıdaki çıktı işe yarayabilir ancak bunu biraz daha görsel hale getirmek isteyebiliriz. Mesela önceki örnekte olduğu gibi, tablo şeklinde tüm özlü sözleri listelemek isteyebiliriz. Bunun için Go kaynak kod dosyasında, dizi kullanacak şekilde bir düzenleme yapmalıyız ve HTML şablon dosyasında da tablo kısmın-da satırları oluşturan <tr>...</tr> kısmını döngüye sokmalıyız.

Aşağıda çok sayıda veriyi web sayfasında tablo halinde görüntüleyen uygulamanın Go ve HTML dosyaları verilmiştir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

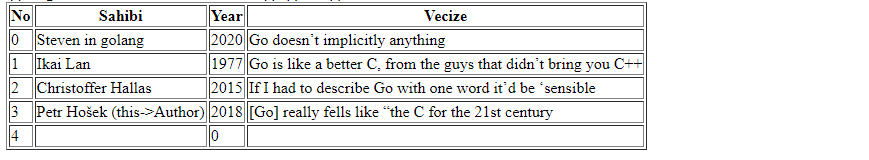
Örnek 3-8-2-2: Template İçerisinde Döngü Kullanımı

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 3-8-2-2: İndex.html Dosyası

Yukarıdaki şablon dosyasında, table bloğunun içerisinde döngü kurulmuştur. Tablonun başlık satırı (<th>...</th>) hariç, diğer satırları (<tr>...</tr>) döngü içerisine dâhil edilmiştir. Program çalıştırıldığında tarayıcı üzerinden görüntülenen sayfa şöyle olmaktadır:



Örnek 3-8-2-2: Tarayıcıdaki Ekran Çıktısı

Tablonun son satırının boş gelmesinin sebebi ise dizi oluşturulurken 5 elemanlı (var unluSozler [5]Söz) belirtilmesi; ancak sadece 4 tane eleman atanmasidır. Dizi 5 elemanlı olduğu için, HTML şablon içerisindeki range fonksiyonu Yine de 5 eleman için ilgili bloğu işletmektedir.

Yönlendirici (router) ifadesi yerine, çoklayıcı anlamına gelen multiplexer (kısaca: mux.) ifadesi de kullanılmaktadır. Web sitesi adreslerinde (URL) http://web\_adresi ------- şeklinde yazılan ifadede web\_adresi/ kısmından sonra gelen kısım alt klasörler için kullanılmaktadır. Bu yapı bazen birkaç tane bölü Işareti de içerebilir. Her bölü işareti bir alt klasörü ifade etmektedir. Ancak yönlendiriciler sayesinde sanal alt klasörler oluşturulabilir. Yönlendiricinin görevi; web adresinin sonuna eklenen kısmı ayrıştırmak ve belirlenen kurallara göre ilgili fonksiyonu çağırmaktır.

Go içinde standart olarak gelen net/http paketi içerisinde zaten yönlendirici işlevi bulunmaktadır. Önceki uygulamalarda bu özelliği zaten kullanılmıştı. Ancak farklı bir yönlendirici paketleri de bulunmaktadır, ihtiyaca göre farklı bir yönlendirici de kullanılabilir. Örneğin hafifsiklet yönlendirici, performans yönlendiricisi basit yönlendirici gibi farklı amaçlarla tasarlanmış çok sayıda paket bulunmaktadır. GitHub üzerinde Go HTTP Router Benchmark ismi ile yayınlanan bir performans test (benchmarking) uygulaması bulunmaktadır. HttpServeMux yönlendiricisi, Go içinde hazır gelmektedir. Bu yönlendirici genel amaçlı olarak geliştirilmiştir. HTTPServeMuxtiair daha hızlı olanların hedef kullanım alanı, performans gerektiren sistemlerdir. HTTPServeMux'tan daha yavaş olan BeeGo ve Martini gibi yönlendiriciler ise bütün bir web çatısı içerisinde gelmektedir. Bu tarz sistemlerin amacı ise web programlama konusunda çok sayıda farklı türde işleri kullanıcı adına kolaylaştırmaktır.

Web API'leri ile veri paylaşmak için kullanılan JSON biçimi, günümüzde neredeyse standart haline geldi. Hemen hemen tüm web dilleri JSON üretebilir ve JSON okuyabilir haldedir. Go içinde de JSON desteği hazır halde gelmektedir. Tek yapılması gereken işlem, "encoding/json" paketinin yüklenmesidir. Aşağıda bununla ilgili bir örnek program verilmiştir. Örnek uygulamada, HTML şablon bölümünde kullanılan özlü sözler yapısı kullanılacaktır. Aynı verileri bu sefer HTML tabİosu biçiminde değil de JSON biçiminde sunalacaktır.

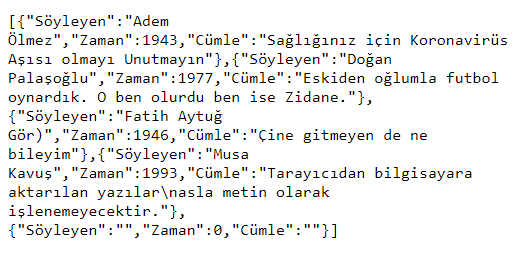


Örnek 3-8-2-3: Json Örneği



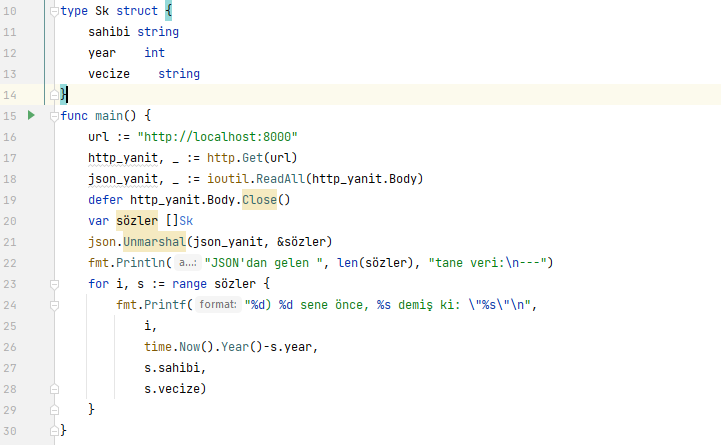
Örnek 3-8-2-3: Devamı

Yukarıdaki programda SözListele fonksiyonunun son satırları, JSON oluşturma ve yayınlama kısmını oluşturuyor. [Json.Marshal](http://json.Marshal) (ÖzlüSözler) ifadesi ile JSON verisi  
oluşturuluyor. Program çalıştırıldığında, tarayıcıdaki ekran görüntüsü aşağıda verilmiştir:



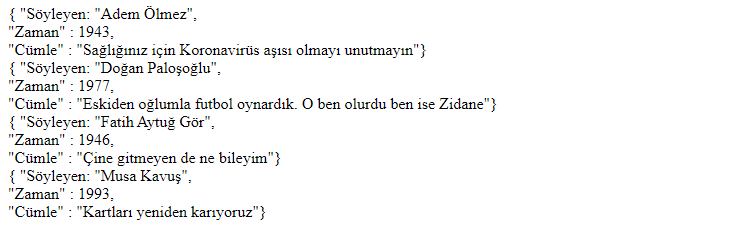
Örnek 3-8-2-3: Ekran Çıktısı

Önceki uygulamanın çıktısını alıp işleyen bir uygulama oluşturulacaktır. LitelDE içerisinde aynı anda iki uygulama çalıştırılmasına izin vermez. Bu nedenle önceki uygulama konsoldan çalıştırabilir. Uygulamayı çalıştırdıktan sonra netstat komutu ile [8000](tel:8000) portunun açık olup olmadığını kontrol etmek işe yarayabilir. Önceki uygulamada JSON verisini oluşturmak için json.Marshal fonksiyonu kullanılmıştır. Bu sefer tam tersi işlem yapılacaktır. Json biçimindeki veriyi parçalarına ayrıştırıp, verileri yapı dizi de değişkenine aktarılacaktır. Bunun için kullanılması gereken fonksiyon, json.Unmarshall fonksiyonudur. Json.Unmarshal(Json.verisi, &yapidegiskeni) kullanım biçimidir.



Örnek 3-8-2-4: Json Okuma Yazma

Json verisini işleyebilmek için öncelikle verinin http üzerinden bilgisayara indirilmesi gerekmektedir. Programda, kendi bilgisayarınızda Json verisini sunan önceki uygulamadan veri çekilerek işlenmektedir. Program çalıştırıldığında şu şekilde çıktı verecektir:



Örnek 3-8-2-4: Ekran Çıktısı

# 4. Örnek Uygulama

## 4.1. Açıklamalı Anlatım

Zaropoly, 1945 yılında ortaya çıkmış bir oyundur. Oyuncular, sırasıyla iki zarı istedikleri kadar atarlar ve o turda gelen sonuçlar toplanır. Eğer zarlardan bir tanesi 1 gelirse kullanıcının o turda aldığı puan 0 kabul edilir ve sıra rakibe geçer. Oyuncu zarın 1 gelme olasığını riske atmamak ve o turda topladığı puanları garantiye almak için durabilir ve sırasını rakibine verebilir. Tüm turlarda kazanılan puanlar devamlı toplanır ve 100 puana ulaşan ilk oyuncu oyunu kazanmış sayılır. Bunun yanı sıra kodlaması yazılacak programda oyuncuların kazanma oranını gösteren veriler de gösterilecektir.

## 4.2. Uygulama Kodları ve Detaylı Açıklama



Örnek 4-1: Paketler

Çalıştırılabilir programlarda yazılan kodların çalıştırılması için **Main Paketi** kullanılır. Main Paketi, Go Derleyicisi’ne diğer paketlerin paylaşılabilir bir kütüphane olduğunu belirtir. Bu paket yazdığımız programda giriş noktası olacaktır.

Bir paketi diğer paketlere ilave etmek için “import” ifadesi kullanılır. Bu programda ekrana yazı yazılması için “fmt” paketi, gelen zar sonucunun rastgele olması için Math Kütüphanesi’nden Rand Paketi ve kullanıcıdan veri almak için Bufio ve Os paketleri kullanılmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

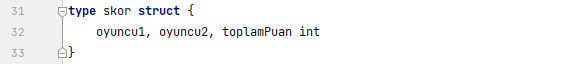
Örnek 4-2: Kullanıcılardan Username Girdisi Almak

Örnek 4-2’ de, loginScreen diye bir fonksiyon tanımlanmış olup parametre atanmamıştır. Bufio Paketi’nden gelen Scanner adlı değişken sayesinde kullanıcıdan veri alımı amaçlanmıştır. Ekrana Username girişi isteyen ekran çıktısı geldikten sonra Scanner.Scan() fonksiyonu sayesinde kullanıcının klavyeye girdiği değerler input sayılacaktır. Daha sonra klavyeye girilen bu değerler oyuncu1 ve oyuncu2 adlı değişkenlere Scanner.Text() fonksiyouyla aktarılacaktır. Bunun yanı sıra oyuncu1 ve oyuncu2 bilgilerini ilerde kullanacağımız için Stringler şeklinde return edilmiştir.



Örnek 4-3: Sabit Tanımı

Const, isminden de anlaşılacağı üzere sabit demektir. Program boyunca galibiyetPuani ve birSeridekiOyun değişkenlerinin değeri değişmeyeceği için sabit olarak tanımlanmıştır. Burda galibiyetPuani, oyunu kazanacak kişinin ulaşacağı puanı temsil eder, birSeridekiOyun ise bir oyuncunun 1 atmadığı sürece oynayacağı maksimum zar atma sayısını temsil eder.



Örnek 4-4: Skor İsimli Struct Tanımı

Nesne Tabanlı Programlama Bölümü’nde anlatıldığı üzere Go Programlama Dili’nde “class” kavramı bulunmamaktadır. Onun yerine ağırlıklı olarak Struct kullanımı tercih edilir. Bu sebeple Örnek 4-4’te “skor” isminde bir Struct oluşturulmuş olup içerisine İnteger tipinden oyuncu1 bir oyuncu2 ve toplamPuan isimli değişkenler atanmıştır. Bu değişkenlerden oyuncu1 ve oyuncu2, oyuncuları temsil ederken toplamPuan ise oyuncuların garanti altına aldıkları toplam puanları temsil etmektedir.



Örnek 4-5: Bazı Değişken Tanımları

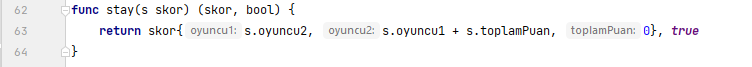
Burda action isimli bir metot tip tanımlanıyor ve bu metot Örnek 4-4’te oluşturulan “skor” adlı Struct’tan kalıtım almaktadır. Bu metod bu Struct’tan “result” adlı bir değeri ve “siraBittiMi” diye bir Boolean değerini döndürmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 4-6: Zarla Metodu

Zarla isimli bir metod tanımlanmış olup bu metod “skor” isimli Struct’tan kalıtım almaktadır. Skor cinsinden s adlı bir değişken oluşturulmuştur ve bu fonksiyon skor ve booelan cinsinden iki değer döndürmektedir. Burda “zardanGelen” isimli değişken zarın her atılışında gelen sayıyı temsil eder ve bu değişken Math Kütüphanesi’nin Rand paketinden 1’den 6’ya kadar random sayı değeri almaktadır. Her zar atışında zardan gelen sonuç fmtPrintln () fonksiyonu sayesinde ekrana yazılmaktadır. Eğer zardan gelen sayı 1 değerine eşit ise o turda alınan toplam puan 0 sayılıp return edilmektedir. Aksi takdirde zardanGelen sayı değişkeni toplam puana ilave edilir. zardanGelen değişkeni ve toplamPuan değişkenleri toplanıp return edilir.



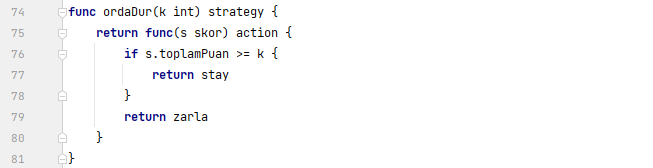
Örnek 4-7: Stay Metodu

Aynı şekilde skor adlı Struct’tan yine kalıtım alınır. Skor ve boolean tipinden return işlemi gerçekleşir. Bu metot önceden oluşturulan result ve siraBittiMi değişkenlerini döndürür. Burda toplamPuan değişkeninden gelen puan oyuncunun ve rakibin skoruna eklenir.



Örnek 4-8: Strategy Metodu

Örnek 4-5’te fonksiyon tipinde “action” isimli bir metot tip tanımlanmıştı. Oluşturulan o tipten “strategy” isimli skor isimli Stuct’tan kalıtım alan bir metot tanımlanıyor.



Örnek 4-9: ordaDur Fonksiyonu

Bu kod bloğunda, ordaDur isimli, İnteger tipinde k isimli bir parametre alan ve strategy tipinden veri döndüren bir fonksiyon tanımlanmıştır. Bu fonksiyon içerisinde return edilecek fonksiyon da kendi içerisinde toplamPuan, k’dan büyük eşit olmak şartıyla ya bir stay Structı döndürecek ya da if koşulu sağlanmadığı takdirde zarla fonksiyonunu return edecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 4-10: oyna Fonksiyonu

Örnek 4-10’de “oyna” isimli bir fonksiyon tanımlanmış olup strategy tipinden strategy0 ve strategy1 değişkenleri parametre olarak atanmıştır. Bu fonksiyonun içerisinde “strategies” isimli bir dizi önceki konularda bahsedildiği üzere Slice Metoduyla oluşturulmuştur. Skor Structı’ndan bir “s“ Nesnesi oluşturulmuş olup Boolean Cinsi’nde siraBittiMi diye bir değişken tanımlanmıştır. Oyuna ilk başlayacak kişinin yerini tutan aktifOyuncu değişkeni Rand fonksiyonuyla belirlenmektedir. Daha sonra bir For Döngüsü oluşturulmuş ve bu döngü 1. Oyuncunun mevcut tüm puanının galibiyetPuanından düşük olması şartıyla çalışacaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 4-11: tekrarTekrar Fonksiyonu

Burda önceden oluşturulmuş “strategies” tipinden “strategy” isimli bir dizi, parametre olarak tekrarTekrar fonksiyonuna atanmıştır. Bu fonksiyon []int tipinden bir dizi ve İnteger bir değer döndürecektir. Sonra “wins” isimli bir Slice tanımlanmış ve boyutu da “strategies” parametresine eşitlenmiştir. 1. Oyuncunun kazanması durumunda ekrana çıktı olarak “1. Oyuncu kazandı” yazısı, 2. Oyuncunun kazanması durumunda ise “2. Oyuncu kazandı” yazısı ekrana çıkacaktır. Döngüler bittikten sonra “wins” isimli dizi ve “gamePerStrategy” adlı İnteger değişkenleri return edilecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 4-12: oran Fonksiyonu

Bu kod bloğunda “oran” ismiyle oluşturulmuş bir fonksiyon görülmektedir. Bu fonksiyon String bir değer döndürecektir. (Vals …int) terimi fonksiyonlarda bahsedildiği üzere ne kadar parametre alacağı belli olmayan fonksiyonlarda kullanılırdı. Bu fonksiyonun kullanılış amacı tamsayı değerlerini alarak oyun esnasında tarafların kazanma ve kaybetme oranını belirlemektir. For döngüsüne değişken geldiği sürece İf Bloğundaki şart kontrol edilecek. Bu fonksiyonda “pct” değişkeni gelen tüm değerlerin toplamının yüzdesini alır. Örnek vermek gerekirse 3 turlu zardan 1, 3, 5 sayıları gelmesi durumunda 1/6 = %16.7, 3/6 gelmesi durumunda %50, ve 5/6 gelmesi durumunda %84,3 gibi bir değere esşit olacaktır. Daha sonra ekrana Fmt.Sprintf () fonksiyonu ile val, total ve pct değişkenleri yazılacaktır. Son olarak s değişkeni return edilecektir.

metin içeren bir resim

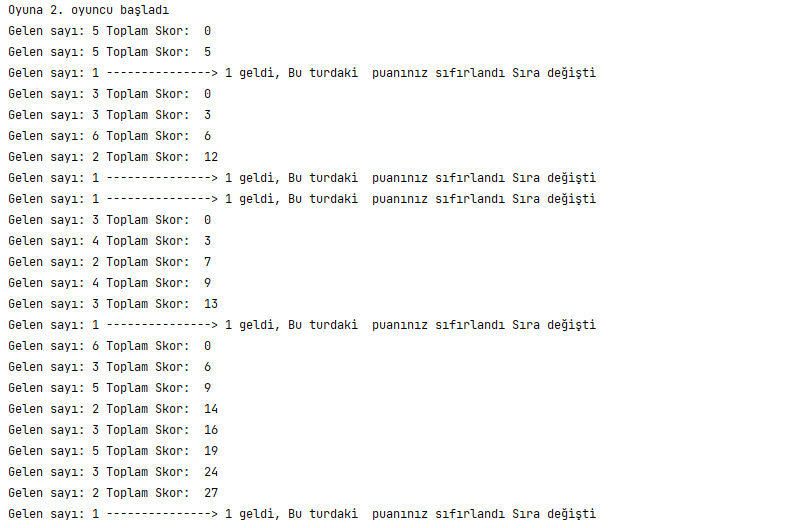
Açıklama otomatik olarak oluşturuldu

Örnek 4-13: main Fonksiyonu

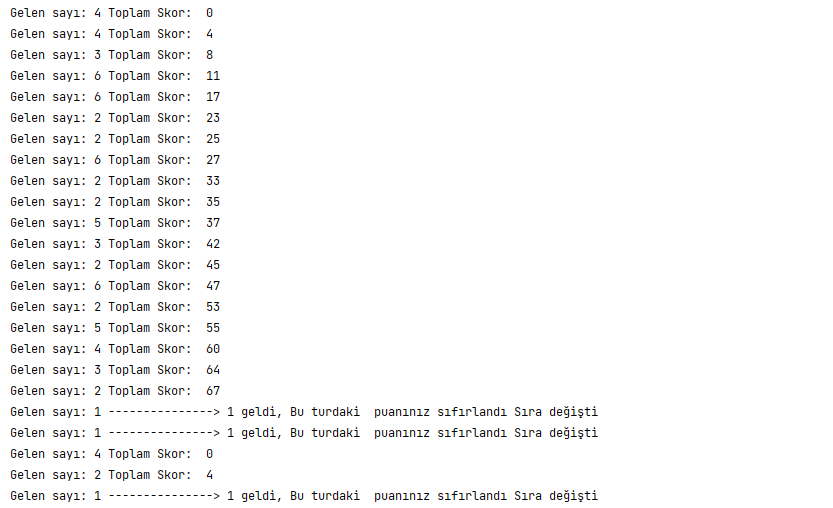
## 4.3. Ekran Çıktıları

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

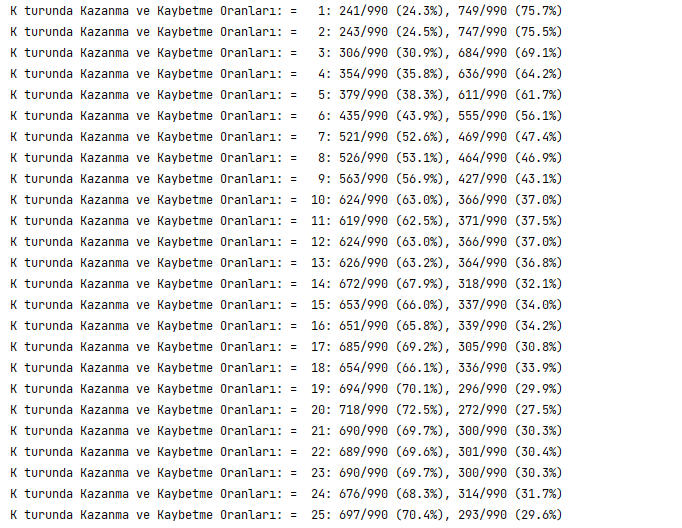
Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

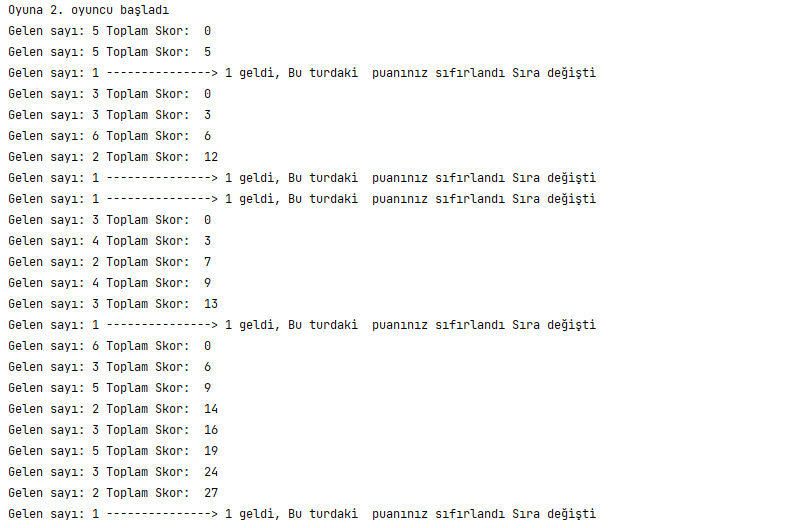
Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

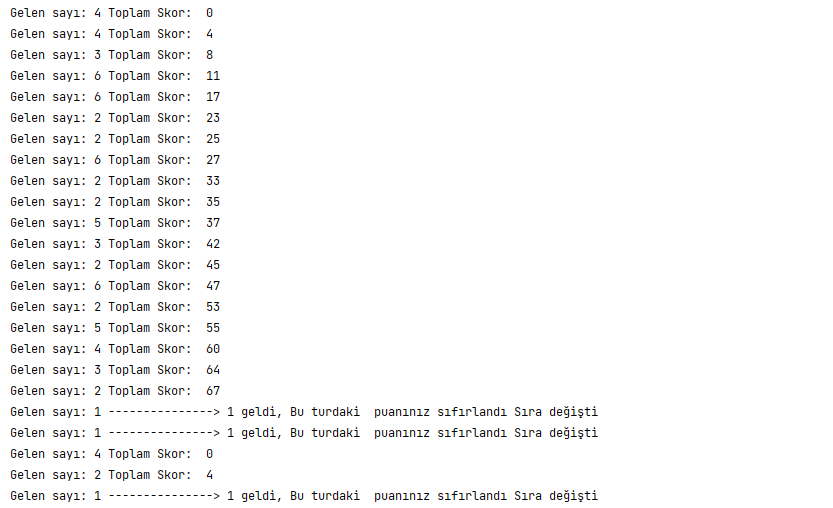
Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

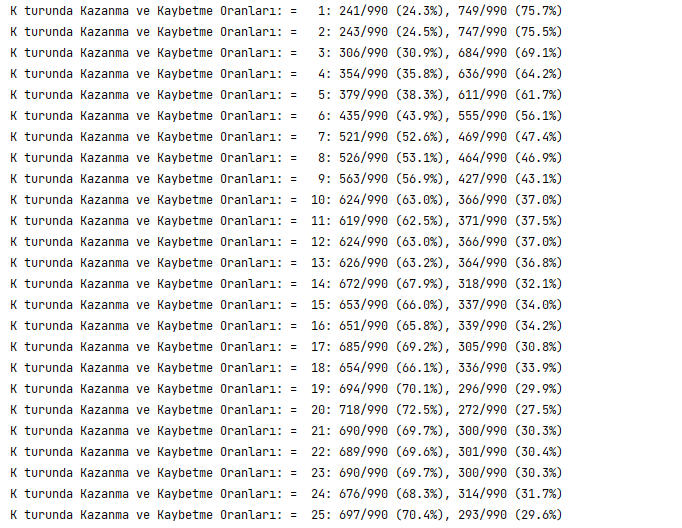
Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

# ****5. Kaynaklar****

(tarih yok).

Charles Thompson. (2017). About Go.

Gerrand, A. (2015). *Go Quotes.*

Özalp, M. (2017). *Go Nasıl Bir Dildir.* İstanbul: Dikeyeksen Yayınları.

Özalp, M. (2017). *Lite İDE Kurulumu.* İstanbul: Dikeyeksen Yayınları.

Supersillyus. (2011, 7 10). Go Rocks - How Can We Avoid Something This Bad In The Future? *Hacker News*, s. 1.

# ****6. Ek Sorular ve Cevaplar****

6.1. **Hangi tür veri tipi bağlama özelliğine sahiptir? Açıklayınız.**

**Go statik tiplidir, gerekirse dinamik tipte de kullanılabilir.** Bir dilin statik veya dinamik tipli olması, dilin karakteristliğini belirleyen önemli bir farktır. Statik dillerde derleme sırasında (Compile Runtime) değişken tip kontrolü yapıldığından, çalışma sırasında (Run-Time) bu işlem için ayrıca emek harcanmaz ve performansı artırır. Dinamik diller ise programcıya birtakım kolaylıklar sağlar. Ne türde verinin geleceği bilinmeyen durumlarda, dinamik tipli bir değişken kullanımı pratik olmaktadır. Ancak dinamik tip kullanımının sorun giderme (debug) durumunda olumsuz etkisi olabilmektedir. Bir kod yığının içinde bir değişkenin tipinin ne olduğu o andaki değerine bağlıdır. Bazen de değişkene farklı tipte veri gelmesi nedeniyle programda analizi güç olan hatalar meydana gelmektedir. Bu sebeble **Go Programlama Dili’nde static tip kullanımı techil edilmiştir. Ancak istendiğinde dinamik değişken kullanma esnekliğide tanınmıştır.**

6.2. **Basit örnek ya da örnekler kullanmak suretiyle hangi tip kapsam özelliklerine sahip olduğunu açıklayınız.**

Golang'da değişkenlerin bildirilebileceği üç yer vardır.

* Yerel değişkenler
* Genel değişkenler
* Formal parametreler

Yerel global değişkenlerin ve Formal parametrelerin ne olduğunu inceleyelim:

**Yerel Değişkenler:** Yerel değişkenler, bir fonksiyon veya blok içinde bildirilenlerdir. Yerel değişkenler, yalnızca o işlev veya kod bloğu içindeki ifadeler tarafından kullanılabilir. Ancak, bu değişkenlere fonksiyon içindeki iç içe kod bloğu ile erişilebilir. Bu değişkenlere blok değişkenler de denir. Bu değişkenler aynı kapsamda aynı adla iki kez bildirilirse derleme zamanı hatası olacaktır. For Döngüsü’nün dışında bildirilen değişkene iç içe döngüler içinde de erişilebilir. Bu, global değişkenin metodlar tarafından erişilebilir olacağı anlamına gelir. Yerel değişken, bir döngü ve bu fonksiyonun içindeki işlev için erişilebilir olacaktır.

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

****

**Global Değişkenler:** Global değişkenler, genellikle programın üstünde, fonksiyonun dışında tanımlanır. Değerlerini programın ömrü boyunca korurlar ve bir program için tanımlanmış herhangi bir fonksiyondan erişilebilirler. Kısaca, fonksiyon veya bir blok dışında tanımlanan değişkenler Global değişkenler olarak adlandırılır. Bunlar bir programın ömrü boyunca kullanılabilir. Global değişkenler, tüm fonksiyonların veya blokların dışında programın en üstünde bildirilir. Global değişkenlere programın herhangi bir bölümünden erişilebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

****

Fonksiyonun içindeki global değişkenle aynı ada sahip yerel bir değişken varsa ne olacak? Cevap basit. Derleyici, yerel bir değişkene tercih verecektir. Genellikle aynı ada sahip iki değişken tanımlandığında, derleyici bir derleme zamanı hatası üretir. Ancak değişkenler farklı kapsamlarda tanımlanmışsa, derleyici buna izin verir. Global bir değişkenle aynı ada sahip bir yerel değişken tanımlandığında, derleyici yerel değişkene öncelik verecektir.

**Formal Parametreler:** Formal parametreler, bu fonksiyon içindeki yerel değişkenler olarak ele alınır ve global değişkenlere göre tercih edilirler.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

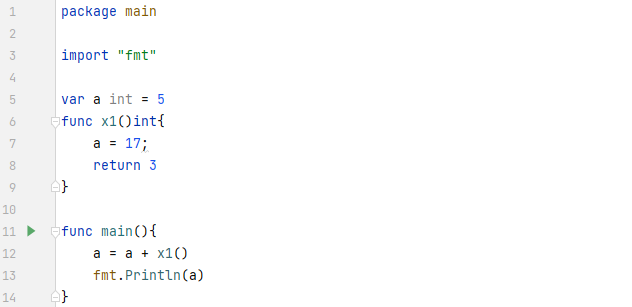
Açıklama otomatik olarak oluşturuldu

Yukarıdaki fonksiyonu çağırdığımızda, global kapsamı değil, yerel kapsamın değerini iletecektir, bu yüzden z'nin çıkış değeri 40 değil 30'dur.

Verilen 3 başlıktan yola çıkarak bir [değişkenin](https://appdividend.com/2020/01/21/keywords-and-identifiers-in-golang-explained/) kapsamı, [tanımlayıcının](https://appdividend.com/2020/01/21/keywords-and-identifiers-in-golang-explained/)[değişken](https://appdividend.com/2020/01/22/golang-variables-example-variables-in-go/) , [sabit](https://appdividend.com/2019/05/16/golang-constants-tutorial-with-example-constants-in-go/) , paket vb. gibi belirli bir değere bağlandığı kaynak kodun bir parçasıdır.

6.3. **Fonksiyon çağrımlarında operatör yan etkisi olup olmadığını bir örnek ile açıklayınız.**

Fonksiyonel yan etki olarak adlandırılan bir fonksiyonun yan etkisi, fonksiyon parametrelerinden birini veya global bir değişkeni değiştirir. X’in a'yı değiştirme yan etkisi yoksa, o zaman değerlendirme sırası iki işlenen, a ve x(a), ifadenin değeri üzerinde hiçbir etkisi yoktur. Ancak x, a'yı değiştirirse, bir etki vardır.

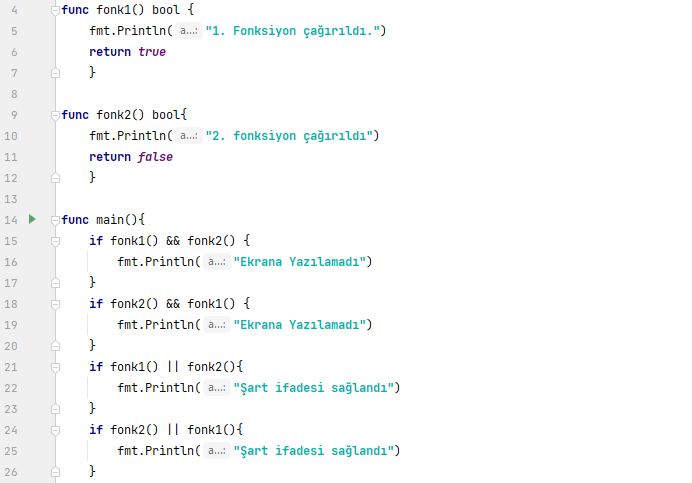


İlk olarak a değeri getirilirse (ifade değerlendirme sürecinde), değeri 10'dur ve ifadenin değeri 20'dir (a + x(a)= 10 + 10). Ancak önce ikinci işlenen değerlendirilirse, ilk işlenenin değeri20'dir ve ifadenin değeri 30'dur (a + x(a)= 20 + 10). Fonksiyonel yan etkiler sorununaiki olası çözüm vardır: Fonksiyonel yan etkilere izin vermemek için dil tanımını yazın. İşlenen değerlendirme sırasının sabitlenmesini talep etmek için dil tanımını yazın. Son olarak Go, işlenenlerin soldan sağa sırayla değerlendirilmesini garanti eder ve bu durumu ortadan kaldırır.

6.4. **Bir örnek içerisinde kısa devre (short-circuit) hesaplama olup olmadığını gösteriniz.**

Go Programlama Dili spesifikasyonu Kısa Devre hakkında net bir görüş belirtmese de şunu ifade eder;

Mantıksal operatörler, boole değerlerine atanır ve işlenenlerle aynı türden sonuç verir. Doğru işlenen koşullu olarak nitelendirilir. Go’nun kısa devre değerlendirmesi yaptığını kanıtlamak için şu örnek verilebilir;



Örnek 6-4: Kısa Devre Örneği



Örnek 6-4: Ekran Çıktısı

6.5. **Fonksiyonlarda parametre geçişi hangi türde olabilmektedir.**

Golang’da fonksiyon terimi, belirli bir görevi yerine getirmek ve sonucu kullanıcıya döndürmek için kullanılan ifadeler topluluğudur. Go parametreleri hem işaretçilerle (referans-pointer) hemde değerlerle (value) geçirmeye izin verir. İkisinin de birbirine göre avantajları ve dezavantajları vardır:

**Parametreler değer (value) olarak iletilirse;**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 6-5-1: Fonksiyona Value Olarak Değer Geçirme

Bir value parametre olarak her geçirildiğinde değişkenin yeni bir kopyası oluşturulur ve çağrılan fonksiyona veya metoda aktarılır. Kopya farklı bir bellek adresinde tahsis edilir. Hemen aşağıda Örnek 6-5’in ekran çıktısında a değerinin test fonksiyonu ile değiştirilemeyeceği görülebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 6-5-1: Ekran Çıktısı

**Parametreler işaretçi (pointer) olarak iletilirse;**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 6-5-2: Fonksiyona Referans Olarak Değer Geçirme

İşaretçi tarafından bir değişkenin iletilmesi durumunda, aynı bellek adresine işaretçinin yeni bir kopyası oluşturulur. Fonksiyona parametre olarak işaretçi girildiği takdirde Örnek 6-5-2 ekran çıktısında görüldüğü değerler değişecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 6-5-2: Ekran Çıktısı

Şunu da ifade etmek gerekir ki Go biraz C Dili’ne benzese de derleyicisi daha farklı çalışır. **Go’da fonksiyona, değere göre parametre geçirmek referansa göre parametre geçirmekten daha ucuzdur.**