



Fast and Easy Programming Language: Go

Mustafa OF*

Kocaeli Üniversitesi, Kocaeli Meslek Yüksekokulu, Bilgisayar Teknolojileri, Kocaeli, Türkiye

Keywords:

Programming Languages, Go Programming Language, Software Development, Web Programming

Abstract

Many people know that the hardware is not working without software. Software is developed by programming languages. One of the most important factors determining the reach of software is the characteristics of the programming language. Many applications can be developed with the Go programming language, a language that is fast, easy to learn and has a large library. The aim of this study is to give information about Go programming language, which is a powerful, fast, easy to learn programming language and to make explanations about basic concepts. Go programming language, developed by Google. Many of the deficiencies of traditional programming languages have been eliminated. It appeared in 2009 and 1.0 version was released in 2012. With the Go programming language, fast and sophisticated projects that can work on the web or in a different environment can be produced. It is an open-source programming language that is evident by the notion of rule, flexibility and speed. In a short time, he was among the fastest growing programming languages.

In this study, advanced aspects of Go programming language compared to other programming languages will be explained. The superior features that make programming easier will be discussed. The understanding of the language will be facilitated by developing sample codes.

Hızlı ve Kolay Bir Programlama Dili: Go

Anahtar Kelimeler:

Programlama Dilleri, Go Programlama Dili, Yazılım Geliştirme, Web Programlama

Özet

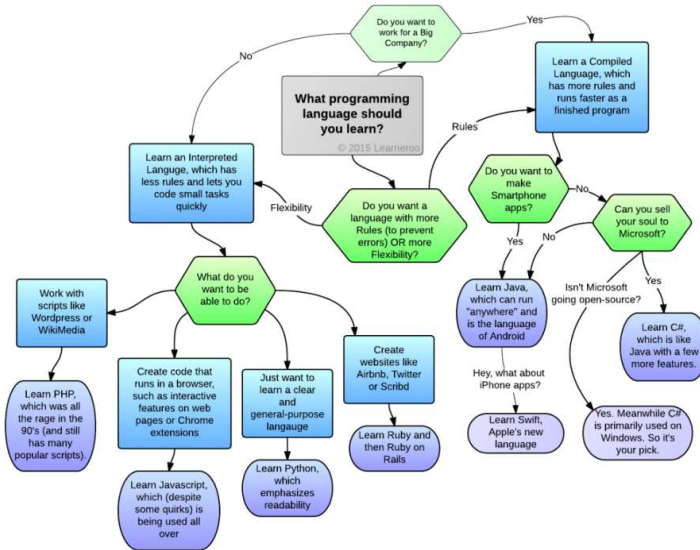
Birçok kişi, donanımın yazılım olmadan bir işe yaramadığını bilir. Yazılım, programlama dilleri ile geliştirilmiştir. Yazılımın gücünü belirleyen en önemli faktörlerden biri, programlama dilinin özellikleridir. Birçok uygulama, hızlı, öğrenilmesi kolay ve geniş bir kütüphaneye sahip bir dil olan Go programlama dili ile geliştirilebilir. Bu çalışmanın amacı, güçlü, hızlı, öğrenilmesi kolay bir programlama dili olan Go programlama dili hakkında bilgi vermek ve temel kavramları hakkında açıklamalarda bulunmaktır. Google tarafından geliştirilen programlama dili olan Go Programlama dili geleneksel programlama dillerindeki eksikliklerin birçoğunu ortadan kaldırmıştır. İlk olarak 2009 yılında çıkmıştır ve 1.0 sürümü 2012 yılında piyasaya sürülmüştür. Go programlama dili ile web üzerinde veya farklı bir ortamda çalışabilecek hızlı ve gelişmiş projeler üretilebilmektedir. Kural, esneklik ve hız kavramıyla ortaya çıkan bir açık kaynak programlama dilidir. Kısa sürede, en hızlı büyüyen programlama dilleri arasına girmiştir.

Bu çalışmada Go programlama dilinin diğer programlama dillerine göre gelişmiş tarafları açıklanmıştır. Programlamayı daha kolay hale getiren üstün özellikleri tartışılmıştır. Örnek kodlar geliştirilerek dilin anlaşılması kolaylaştırılmıştır.

1 GİRİŞ

Go dili başlangıçta 2007'de Robert Griesemer, Rob Pike ve Ken Thompson tarafından geliştirilen bir programlama diliydi. Daha sonra Google tarafından geliştirilmeye başlanmıştır. Bu dil, statik bir yazılı dildir ve C programlama diline benzer bir söz dizimine sahiptir. Çöp toplama, tip güvenliği, dinamik kodlama, değişken uzunluklu diziler ve anahtar-değer haritaları gibi birçok gelişmiş yerleşik tür bulunmaktadır. Aynı zamanda zengin bir standart kütüphaneye sahiptir. Go programlama dili Kasım 2009'da piyasaya sürüldü ve Google'ın bazı sistemlerinde kullanılmaktadır.

Google, büyük bir teknoloji ve şirkettir ve yıllardır C, C ++, Java, Python vb. birçok farklı programlama dili kullanmaktadır. Bu teknolojilerin çeşitli avantaj ve dezavantajları vardır. Düşük veya az yük altındaki yazılım projelerinde bunlar fark edilmeyebilir, ancak büyük projelerde ortaya çıkarlar. Bunlar performans, derleme (büyük projelerde kaynak kodunu derlemek zaman alabilir) güvenlik, uyumluluk, zaman yönetimi, kaynaklar (donanım, enerji vb.), bakım yönetimi gibi birçok başlıkta listelenebilir. Google, yıllardır hem iç sistemler için bir işletim sistemi, hem de bunun gibi sorunları çözmek için birçok teknoloji ve algoritma geliştirdi. Go programlama dili de bunlardan biridir. Go programlama dili, sorunları çözmek için Google tarafından çıkarıldı. Bu nedenle, Go'ya eklenen veya eklenmeyen tüm özellikler, yılların yazılım deneyimleri ile belirlenmiştir. Go programlama dilini biraz incelediğinizde, "Neden jenerik değil?" gibi bir soru aklınıza gelebilir. Bu soruyu geliştirici ekibinden biri şöyle cevaplamaktadır; "Jenerikler hızlı değildir". Go'nun bakış açısı çok açık bir şekilde az dil yeteneği ve kuralı ile esnek, hızlı ve güçlü bir dil oluşturmaktır. Bu çalışmada, Go programlama dilinin temel özellikleri açıklanacaktır. Yazılım dünyasına getirdiği kolaylıklar tartışılacaktır. Örnek kodlar verilecektir. Hızlı bir dil olan GoLang, temel hatları ile incelenecektir. Bu makalenin oluşturulması esnasında kullanılan derleyici 1.13.4'tür.



Şekil 1: Hangi programlama dilini öğrenmeli? (Learneroo, 2015)

2 MATERYAL VE METOD

Bir programlama dili ile kod yazmaya başlamak için öncelikle dilin yazım kuralları ve çalışma standartlarının öğrenilmesi gereklidir. Programlama dilleri, İngilizce, Almanca gibi yabancı dilleri öğrenmeye çok benzerler. Bir dili öğrendikten sonra diğer dile aşinalık oldukça fazla olacaktır. Temel yazım kalıpları ile başlanılır. Daha sonra programlama dilinin alt yapısını teşkil eden değişkenler, operatörler, veri yapıları gibi yapılarla devam edilir. Bu çalışmada öncelikle Go dilinin temel özellikleri belirtilmiştir. Daha sonra veri yapıları üzerinde durulmuştur. Veri yapıları ile ilgili uygulamalar geliştirilmiş (Kodlanmış) ve derlenerek nasıl bir sonuç ürettiği ortaya çıkartılmıştır.

2.1 Go Programlama Dilinin Özellikleri

Go programlama dilinin önemli özellikleri aşağıda listelenmiştir;

- Derleme süresi hızlıdır.
- Dinamik dillere benzer kalıpları benimseyen ortam desteği. (Örnek: $b = 10$, int türünde bir b değişkeninin tanımlanması)

- GoLang programları basit ve güvenlidir.
- GoLang, nesneye yönelik bir programlama dilidir.

GoLang programlama dilinin basit ve hızlı olmasını sağlayan ve diğer programlama dillerinde olan bazı özellikler bu dilden çıkarılmıştır. Bu özellikler;

- Tip kalıtımı,
- Metot ve operatör için aşırı yükleme (Overloading),
- Paketler arasında genel bağımlılıklar,
- Pointer aritmetiği,
- Jenerik (Generic) programlama

```
package main

import "fmt"

func main() {
    fmt.Println("Merhaba Dünya.")
}
```

Şekil 2: “Merhaba Dünya” yazdıran Go uygulaması

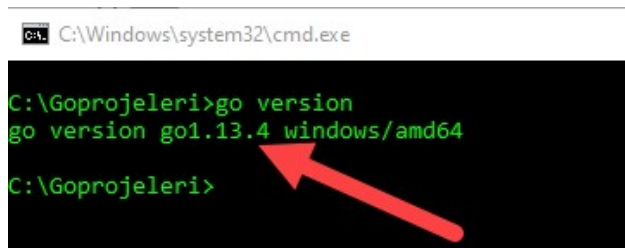
Bir Go programın satır sayısı, 4 satırdan milyonlarca satıra kadar büyüyebilir. ".go" uzantısıyla bir veya daha fazla metin dosyasına yazılabilir. Örneğin, ilkuyg.go. Bir Go programı, Windows işletim sisteminde Not Defteri veya Notepad++, Linux'ta Nano, MacOS'ta TextEdit gibi düzenleme yazılımları kullanılarak oluşturulabilir. Derleme işlemini gerçekleştirmek için gerekli olan tüm paketler www.golang.org adresinden indirilmesi gereklidir. Go dili için farklı IDE (Integrated Development Environment) geliştirme ortamları kullanılabilir. Oldukça popüler olan Eclipse (www.eclipse.org) uygulaması ile kodlar yazılabilir. İstenirse IDE olmadan da Go uygulamaları yazılabilir. Fakat program büyüdükçe kontrolü zorlaşacaktır.

Bir derleyici, bir programlama dilinde (kaynak dil) yazılmış bilgisayar kodunu başka bir programlama diline (hedef dil) dönüştüren bir bilgisayar yazılımıdır. Derleyiciler, başta bilgisayar olmak üzere dijital cihazları destekleyen bir dönüştürme türüdür. Programın, derleme (Compile) işleminden sonra tek başına çalışabilir hale gelmesi için bağlayıcı (Linker) adı verilen yazılımlar kullanılır. Sonuç olarak kendi başına çalışabilen bir yazılım ortaya çıkar. Bu işlemler ayrı olarak yapılabildiği gibi, bütünsel geliştirme ortamlarında (IDE) tek bir tuşla da yapılabilir.

Go derleyicisinin 1.13.4 sürümü aşağıdaki adresten indirilebilir. İşletim sistemine uygun olan derleyici seçimi yapılır ve kurulum sonrasında gerekli derleme ve bağlama alt yapıları sisteme kurulmuş olur.
<https://golang.org/dl/>

Windows işletim sisteminde Go derleyicisinin kurulumu;

Derleyici paketi indirilip kurulumu başlatılır. Kurulum klasörü verildikten sonra gerekli olan tüm ortam değişkenleri hazır hale gelir. Go derleyicisinin kurulup kurulmadığını aşağıdaki gibi kontrol edebiliriz.



```
C:\Windows\system32\cmd.exe

C:\Goprojeleri>go version
go version go1.13.4 windows/amd64

C:\Goprojeleri>
```

Şekil 3: Windows komut satırında Go derleme sürümünün kontrolü

İkinci ve daha alt seviye başlıklar 10 pt. ve kalın harflerle yazılmalıdır. Tablolardan, şekillerden ve alt seviye başlıklardan önce ve sonra da birer satır boşluk bulunmalıdır. Tablo ve şekil açıklamaları sayfaya ortali şekilde hizalanmalıdır. Tablo içi metinlerin büyüklüğü için herhangi bir kısıtlama bulunmamaktadır. Yukarıdaki ekranı görüyorsanız artık Windows'ta Go programlarını yazmaya hazırsınız.

Linux işletim sisteminde kurulumu;

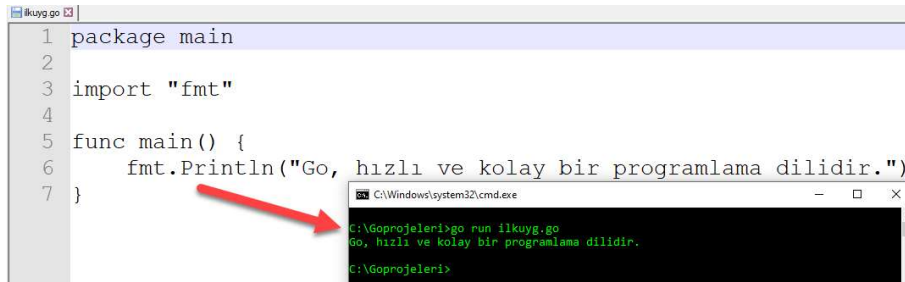
```
$ wget https://dl.google.com/go/go1.13.4.linux-amd64.tar.gz
$ sudo tar -xvf go1.13.4.linux-amd64.tar.gz
$ sudo mv go /usr/local
```

Ortamı hazırlama;

```
GOROOT, Go paketlerinin yüklü olduğu klasördür
$ export GOROOT=/usr/local/go
GOPATH, çalışma klasörüdür. Örneğin çalışma klasörü : ~/GoProjects
$ export GOPATH=$HOME/GoProjects
PATH, ortam değişkeni
$ export PATH=$GOPATH/bin:$GOROOT/bin:$PATH
Komut satırından Go sürümünü kontrol etme
$ go version
go version go1.11 linux/amd64
Artık Linux'ta Go uygulamaları yazmaya hazırsınız.
Kurulum yaptıktan sonra GoLang belgelerine aşağıdaki şekilde ulaşılabilir.
godoc -http=:8000 (Komut satırından yazılır. Komut satırı kapatılmamalıdır.)
```

Web tarayıcı açılarak adres kısmına aşağıdaki bilgi yazılır. Bu sayede GoLang ile ilgili yardım belgelerine ulaşılabilir.
http://localhost:8000

2.2 İlk Go Uygulaması



Şekil 4: Notepad++ uygulamasında ilk programın yazılımı ve derlenip çalıştırılması

Windows işletim sisteminde ilk Go uygulamasını yaparken aşağıdaki adımlar takip edilebilir.

- Metin editörü (Notepad++) açılır ve kodlar yazılır.
- Uygulama `ilkuyg.go` adı ile kayıt edilir.
- Komut satırına geçilir (`cmd.exe` ile)
- Uygulamanın yazıldığı klasöre geçilir.
- Komut satırından `go run ilkuyg.go` yazılır.

2.3 Go Programlama Dili Kuralları

Her dilde olduğu gibi bu dilde de uyulması gereken bazı kurallar vardır. Bunlardan bazıları aşağıya çıkarılmıştır.

Satır Bitimi;

Satır bitimi için herhangi bir karaktere ihtiyaç yoktur.
`fmt.Println("Merhaba")`
`fmt.Println("Go Programlama Dili")`

Açıklamalar;

Derleyici tarafından derlenmeyen yorumlar, program hakkında bilgiler sunar. Ayrıca derlenmesi istenilmeyen kodların ayrı tutulmasını sağlar. Açıklamalar `/* */` karakterleri içerisine alınmalıdır.
`/* Bu bir açıklamadır */`

Tanımlayıcılar;

Bir değişkeni veya fonksiyonu tanımlamak için bir isme ihtiyaç vardır. Tanımlayıcılar, bir harf ile başlamalıdır. Noktalama işaretleri barındırmamalıdır. Sayı ile (0-9) devam edebilirler. Go programlama dili büyük/küçük harf duyarlıdır. Örneğin “okul” ile “Okul” tanımlayıcısı birbirinden farklıdır.

```
bolum, bolum1, bolum2
bolum1 = "Bilgisayar"
bolum2 = "Muhasebe"
```

3 TEMEL VERİ TİPLERİ

Bu kısımda Go programlama dilinin temelini oluşturan veri tipleri açıklanacaktır.

3.1 Değişken Tipleri

Go programlama dilinde kullanılacak değişkenler öncelikle tanıtımı yapılmalıdır. Temel olarak dört tip bulunmaktadır.

- Sayısal
- Metin (String)
- Mantıksal
- Türetilmiş

Sayısal tipler;

Integer tipler (Tamsayı)

```
uint8   İşaretsiz 8 bit integer (0 - 255)
uint16  İşaretsiz 16 bit integer (0 - 65535)
uint32  İşaretsiz 32 bit integer (0 - 4294967295)
uint64  İşaretsiz 64 bit integer (0 - 18446744073709551615)
int8    İşaretili 8-bit integer numbers (-128 - 127)
int16   İşaretili 16-bit integer numbers (-32768 - 32767)
int32   İşaretili 32-bit integer numbers (-2147483648 - 2147483647)
int64   İşaretili 64-bit integer numbers (-9223372036854775808 - 9223372036854775807)
```

Gerçek Sayılar (Real);

```
float32   32 bit real sayılar
float64   64 bit real sayılar
complex64 Kompleks sayılar (float32)
complex128 Kompleks sayılar (float64)
```

Yukarıdaki sayısal tiplerin dışında; byte (uint8 gibi), rune (int32 gibi) uint (32/64 bit), int (uint gibi) ve uintptr (Bir pointer değerinin işaretsiz tamsayı kısmı) tipleri de bulunmaktadır.

Tip tanıtımı şu şekilde yapılır;

```
var değişken_adi değişken_tipi
```

Örnek;

```
var a,b,c int
```

```
var adet, toplam float32
```

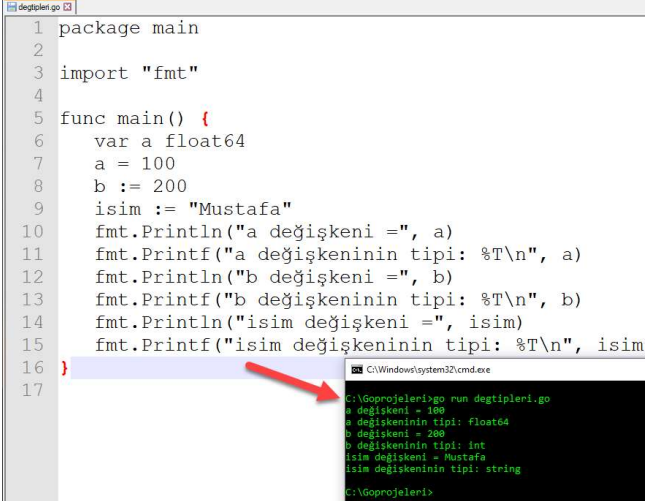
```
var gun = 30 /*İlk değeri verilebilir.*/
```

```
a := 50 /*Tanıtım yapılmadan değeri verilebilir. Burada a, bir int değişkendir.*/
```

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var a float64
7     a = 100
8     b := 200
9     isim := "Mustafa"
10    fmt.Println("a değişkeni =", a)
11    fmt.Printf("a değişkeninin tipi: %T\n", a)
12    fmt.Println("b değişkeni =", b)
13    fmt.Printf("b değişkeninin tipi: %T\n", b)
14    fmt.Println("isim değişkeni =", isim)
15    fmt.Printf("isim değişkeninin tipi: %T\n", isim)
16 }
17

```



Şekil 5: Tip tanıtımı uygulaması

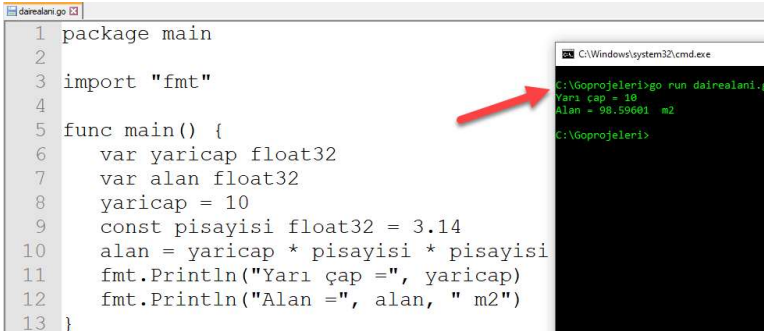
Sabit değişkenlerin tanıtımı;
const değişken_adi tipi = değeri

Örnek;
const yukseklik int =100
const pisayisi float32 = 3.14

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var yaricap float32
7     var alan float32
8     yaricap = 10
9     const pisayisi float32 = 3.14
10    alan = yaricap * pisayisi * pisayisi
11    fmt.Println("Yarı çap =", yaricap)
12    fmt.Println("Alan =", alan, " m2")
13 }

```



Şekil 6: Sabit değişken kullanımı uygulaması

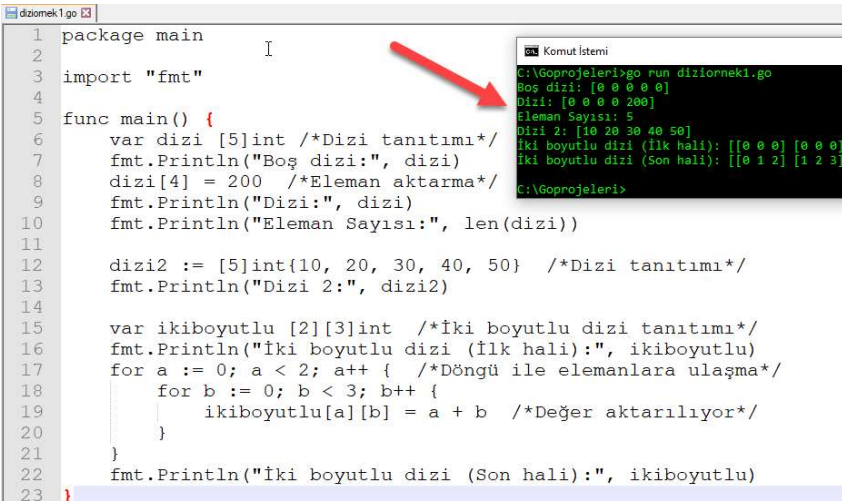
3.2 Diziler

Birden fazla değeri bir değişken içerisinde barındıran diziler (Array), Go programlama dilinde de oldukça etkin bir şekilde kullanılır. Aşağıda dizilerin kullanımına dair bir örnek verilmiştir.

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var dizi [5]int /*Dizi tanıtımı*/
7     fmt.Println("Boş dizi:", dizi)
8     dizi[4] = 200 /*Eleman aktarma*/
9     fmt.Println("Dizi:", dizi)
10    fmt.Println("Eleman Sayısı:", len(dizi))
11
12    dizi2 := [5]int{10, 20, 30, 40, 50} /*Dizi tanıtımı*/
13    fmt.Println("Dizi 2:", dizi2)
14
15    var ikiboyutlu [2][3]int /*iki boyutlu dizi tanıtımı*/
16    fmt.Println("İki boyutlu dizi (İlk hali):", ikiboyutlu)
17    for a := 0; a < 2; a++ { /*Döngü ile elemanlara ulaşma*/
18        for b := 0; b < 3; b++ {
19            ikiboyutlu[a][b] = a + b /*Değer aktarılıyor*/
20        }
21    }
22    fmt.Println("İki boyutlu dizi (Son hali):", ikiboyutlu)
23 }

```



Şekil 7: Dizilerin kullanımına dair bir örnek

3.3 Dilimler (Slices)

Dilimler, dizilere göre daha geniş bir ara yüz desteği sağlar. Dizilerin aksine, dilimler yalnızca içerdikleri öğeler tarafından yazılırlar. Sıfır olmayan uzunluğu olan boş bir dilim oluşturmak için make deyimi kullanılır. Kullanışlı ek fonksiyonları bulunmaktadır. Aşağıda dilimlerin kullanımına dair bir örnek verilmiştir.

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     dilim1 := make([]string, 3)
7     fmt.Println("Dilim 1:", dilim1)
8     dilim1[0] = "Go"
9     dilim1[1] = "Programlama"
10    dilim1[2] = "Dili"
11    fmt.Println("Dilim 1:", dilim1)
12    fmt.Println("Üçüncü Eleman:", dilim1[2])
13    fmt.Println("Eleman Sayısı:", len(dilim1))
14    dilim1 = append(dilim1, "çok") /*Yeni eleman ilave etme*/
15    dilim1 = append(dilim1, "güçlü", "bir", "dildir")
16    fmt.Println("Dilim 1:", dilim1)
17    dilim2 := make([]string, len(dilim1)) /*Var olan dilimden yeni bir dilim oluşturuluyor.*/
18    copy(dilim2, dilim1) /*Kopyasını alır.*/
19    fmt.Println("Dilim 2:", dilim2)
20    dilim3 := dilim1[2:5] /*2. ve 5. konum arasındakileri alır.*/
21    fmt.Println("Dilim 3:", dilim3)
22    dilim3 = dilim1[:5] /*5. Konuma kadar olanı alır.*/
23    fmt.Println("Dilim 3:", dilim3)
24    dilim3 = dilim1[2:] /*2. Konumdan sonrakileri alır.*/
25    fmt.Println("Dilim 3:", dilim3)
26    gunler := []string{"Psi", "Sal", "Çar", "Per", "Cum", "Csi", "Paz"}
27    fmt.Println("Günler:", gunler)
28 }

```

Şekil 8: Dilimlerin (Slices) kullanımına dair bir örnek

3.4 Haritalar (Maps)

Haritalar, diğer dillerdeki Hasht tablolarına veya sözlüklere (Dictionary) benzetilebilir. Bu yapıda bir anahtar (Key) ve bir değer (Value) ikilisi bulunmaktadır. Anahtar verilerek değere ulaşılabilir. Aşağıda haritaların kullanımına dair bir örnek verilmiştir.

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     /*Anahtar : string, Değer: string*/
7     urunler := make(map[string]string)
8     urunler["Defter"] = "10 Tl"; /*Değerler veriliyor.*/
9     urunler["Kitap"] = "20 Tl";
10    urunler["Kalem"] = "5 Tl";
11    fmt.Println("Ürünler:", urunler)
12    kitapfiyat := urunler["Kitap"] /*Anahtarı vererek değer alınıyor.*/
13    fmt.Println("Kitabın fiyatı: ", kitapfiyat)
14    fmt.Println("Eleman Sayısı:", len(urunler))
15    delete(urunler, "Kalem") /*Eleman silme*/
16    fmt.Println("Ürünler (Yeni Hali):", urunler)
17    /*Tanıtma ve değer aktarma*/
18    ogrenci := map[string]int{"Ali Aslan": 554433, "Ahmet Eren": 665544}
19    fmt.Println("Öğrenciler:", ogrenci)
20 }

```


Şekil 9: Haritaların (Maps) kullanımına dair bir örnek

Aşağıda, range ifadesini kullanarak harita türü değişkenlerin listelenmesi kolaylaştırılmıştır.

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     isimler := map[string]string{"M": "Mustafa", "C": "Celal"}
7     for anahtar, deger := range isimler {
8         fmt.Printf("%s -> %s\n", anahtar, deger)
9     }
10    for anahtar, deger := range isimler {
11        fmt.Println("Anahtar:", anahtar, " Değer:", deger)
12    }
13    /*A ve B değerlerinin Unicode kodunu elde eder.*/
14    for a, b := range "AB" {
15        fmt.Println(a, b)
16    }
17 }

```



The terminal output for the program in Figure 10 is as follows:

```

C:\Goprojeleri>go run haritarange.go
M -> Mustafa
C -> Celal
Anahtar: M Değer: Mustafa
Anahtar: C Değer: Celal
0 65
1 66
C:\Goprojeleri>

```

Şekil 10: Haritalar (Maps) ile range deyiminin kullanımına dair bir örnek

4 PROGRAM AKIŞ DEYİMLERİ

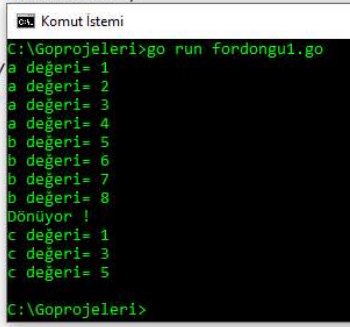
4.1 Döngüler

Her programlama dilinde olduğu gibi Go dilinde de döngüler bulunmaktadır. Temel döngü deyimidir. Aşağıda örnek bir döngü deyimini verilmiştir. Bir veya birden fazla deyim bir döngü içerisinde oluşan döngü sayısı kadar çalıştırabiliriz. Döngüler, programlama dillerinin temel yapı taşlarından biridir.

```

1 package main
2 import "fmt"
3 func main() {
4     a := 1
5     for a <= 4 { /*Sadece başlangıç değeri verilebilir.*/
6         fmt.Println("a değeri=", a)
7         a = a + 1
8     }
9     for b := 5; b <= 8; b++ { /*Standart kullanım.*/
10        fmt.Println("b değeri=", b)
11    }
12    for { /*Parametresiz sonsuz bir döngü*/
13        fmt.Println("Dönüyor !")
14        break /*Döngüden çıkış*/
15    }
16    for c := 0; c <= 6; c++ {
17        if c%2 == 0 { /*Tek sayılar*/
18            continue
19        }
20        fmt.Println("c değeri=", c)
21    }
22 }

```



The terminal output for the program in Figure 11 is as follows:

```

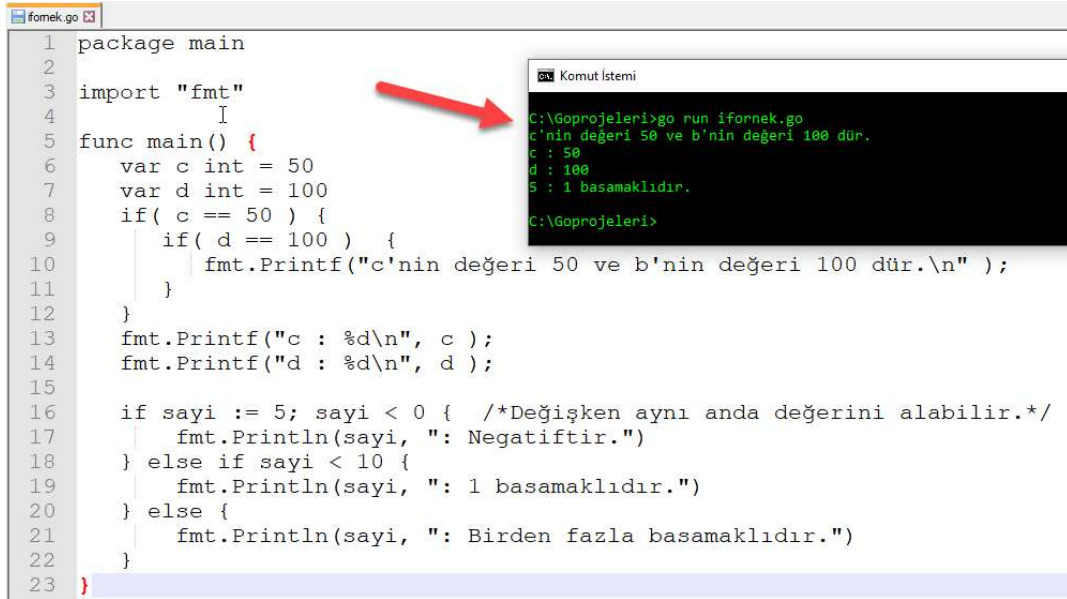
C:\Goprojeleri>go run fordongu1.go
a değeri= 1
a değeri= 2
a değeri= 3
a değeri= 4
b değeri= 5
b değeri= 6
b değeri= 7
b değeri= 8
Dönüyor !
c değeri= 1
c değeri= 3
c değeri= 5
C:\Goprojeleri>

```

Şekil 11: Döngü kullanımına dair bir örnek

4.2 If Karar Deyimi

Karar deyimlerinden if, bilindiği gibi en çok kullanılan bir deyimdir. Mantıksal ifadenin durumuna göre belirlenen işlemleri yapmaktadır. Aşağıda if kullanımına dair bir örnek verilmiştir.



```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var c int = 50
7     var d int = 100
8     if( c == 50 ) {
9         if( d == 100 ) {
10            fmt.Printf("c'nin değeri 50 ve b'nin değeri 100 dür.\n" );
11        }
12    }
13    fmt.Printf("c : %d\n", c );
14    fmt.Printf("d : %d\n", d );
15
16    if sayi := 5; sayi < 0 { /*Değişken aynı anda değerini alabilir.*/
17        fmt.Println(sayi, ": Negatiftir.")
18    } else if sayi < 10 {
19        fmt.Println(sayi, ": 1 basamaklıdır.")
20    } else {
21        fmt.Println(sayi, ": Birden fazla basamaklıdır.")
22    }
23 }

```

```

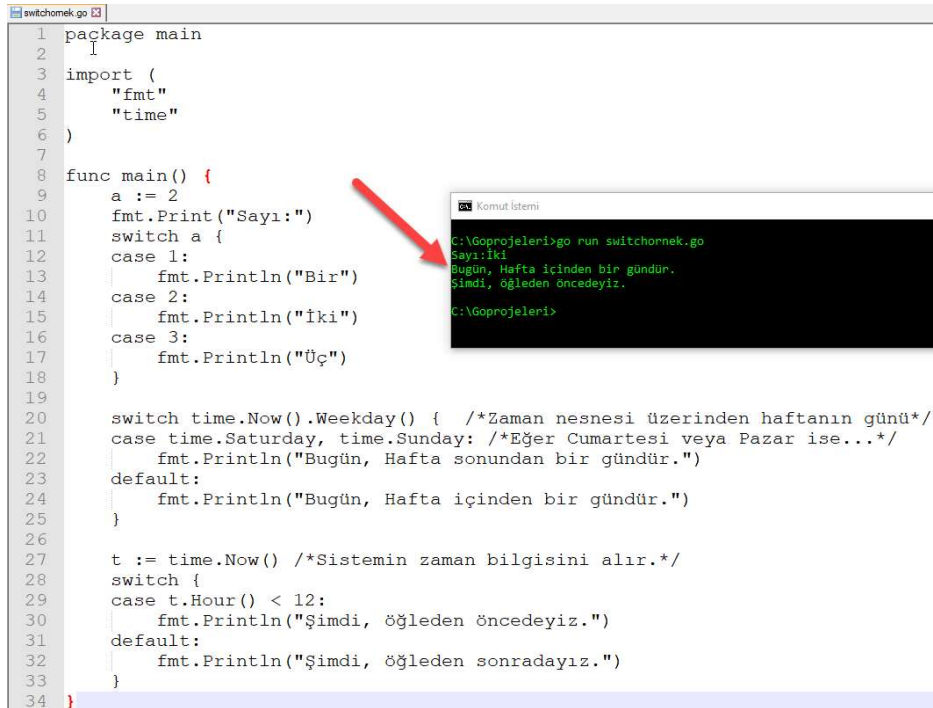
Komut İstemi
C:\Goprojeleri>go run ifornek.go
c'nin değeri 50 ve b'nin değeri 100 dür.
c : 50
d : 100
5 : 1 basamaklıdır.
C:\Goprojeleri>

```

Şekil 12: If karar deyimi kullanımına dair bir örnek

4.3 switch Karar Deyimi

İç içe if-else deyiminin bir benzeri olan switch ile daha kolay karar verme işlemi yapılabilir. Kontrol edilecek değişken üzerinden aldığı değerlere göre program yönlendirilir. Aşağıda switch kullanımına dairi bir örnek verilmiştir.



```

1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 func main() {
9     a := 2
10    fmt.Print("Sayı:")
11    switch a {
12    case 1:
13        fmt.Println("Bir")
14    case 2:
15        fmt.Println("İki")
16    case 3:
17        fmt.Println("Üç")
18    }
19
20    switch time.Now().Weekday() { /*Zaman nesnesi üzerinden haftanın günü*/
21    case time.Saturday, time.Sunday: /*Eğer Cumartesi veya Pazar ise...*/
22        fmt.Println("Bugün, Hafta sonundan bir gündür.")
23    default:
24        fmt.Println("Bugün, Hafta içinden bir gündür.")
25    }
26
27    t := time.Now() /*Sistemin zaman bilgisini alır.*/
28    switch {
29    case t.Hour() < 12:
30        fmt.Println("Şimdi, öğleden öncedeyiz.")
31    default:
32        fmt.Println("Şimdi, öğleden sonradayız.")
33    }
34 }

```

```

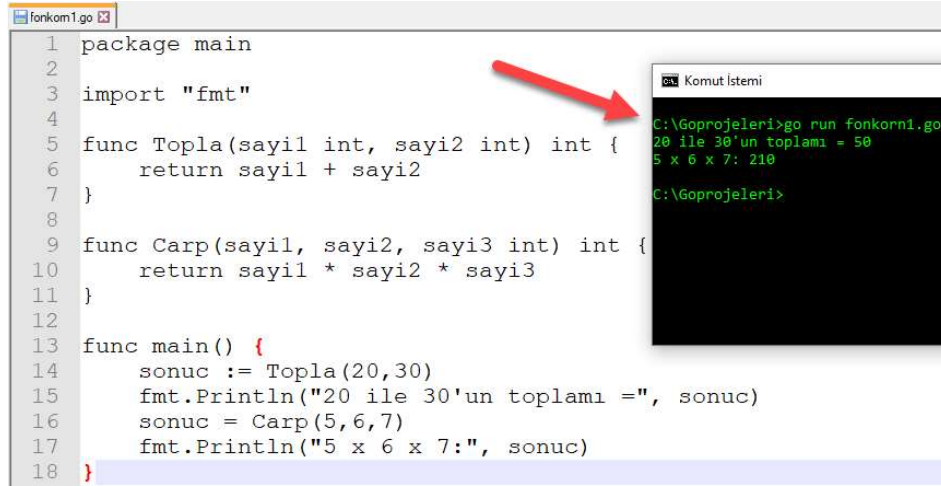
Komut İstemi
C:\Goprojeleri>go run switchornek.go
Sayı:İki
Bugün, Hafta içinden bir gündür.
Şimdi, öğleden öncedeyiz.
C:\Goprojeleri>

```

Şekil 13: switch karar deyimi kullanımına dair bir örnek

4.4 Fonksiyonlar (Functions)

Temel özellikleri bakımından fonksiyonları, ana programdaki akışı farklı bir alt programa yönlendiren yapılardır. Tekrar kullanılması, merkezi bir control sağlaması ve nesneye yönelik programlamanın alt yapısını oluşturması bakımından fonksiyonlar oldukça önemli bir yere sahiptirler. Fonksiyonlar, Go dilinde “func” ile başlarlar. Aşağıda fonksiyon kullanımına dair bir örnek verilmiştir.



```

1 package main
2
3 import "fmt"
4
5 func Topla(sayil int, sayi2 int) int {
6     return sayil + sayi2
7 }
8
9 func Carp(sayil, sayi2, sayi3 int) int {
10    return sayil * sayi2 * sayi3
11 }
12
13 func main() {
14    sonuc := Topla(20,30)
15    fmt.Println("20 ile 30'un toplamı =", sonuc)
16    sonuc = Carp(5,6,7)
17    fmt.Println("5 x 6 x 7:", sonuc)
18 }

```

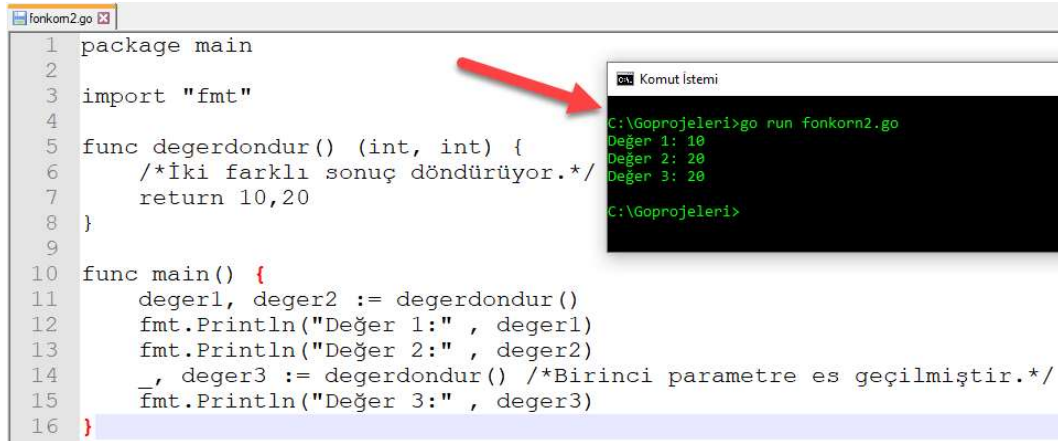
```

C:\Goprojeleri>go run fonkorn1.go
20 ile 30'un toplamı = 50
5 x 6 x 7: 210
C:\Goprojeleri>

```

Şekil 14: Fonksiyon kullanımına dair bir örnek

Fonksiyonlarda genellikle bir adet değer döndürülür. Birden fazla değer döndürmek için programlama dillerinde farklı teknikler kullanılır. Fakat bunu Go dilinde yapmak oldukça kolaydır. Aşağıdaki örnek, birden fazla değer döndürülmesini sağlamaktadır.



```

1 package main
2
3 import "fmt"
4
5 func degerdondur() (int, int) {
6     /*İki farklı sonuç döndürüyor.*/
7     return 10,20
8 }
9
10 func main() {
11    deger1, deger2 := degerdondur()
12    fmt.Println("Değer 1:" , deger1)
13    fmt.Println("Değer 2:" , deger2)
14    _, deger3 := degerdondur() /*Birinci parametre es geçilmiştir.*/
15    fmt.Println("Değer 3:" , deger3)
16 }

```

```

C:\Goprojeleri>go run fonkorn2.go
Değer 1: 10
Değer 2: 20
Değer 3: 20
C:\Goprojeleri>

```

Şekil 15: Birden fazla değer döndüren Fonksiyon kullanımına dair bir örnek

5 SONUÇLAR

Go programlama dili oldukça basit ve kullanışlıdır. Tamamen açık kaynaklıdır. Hızlı bir derleme yapısı bulunmaktadır. Bu durum büyük verilerle yapılan işlem hızını artırır. Standart kütüphanesi oldukça geniş olan ve her geçen gün büyüyen Go programlama dili, yeni bir programlama dili arayan programcılar için iyi bir seçimdir. Diğer dillerde olmayan bir çok kolaylaştırıcı metot, Go dilinde bulunmaktadır. Bu makalede, Go programlama dilinin temel özellikleri açıklanmıştır. Bu dil yeni programcılar için farklı bir başlangıç oluşturacaktır. Var olan programlama dillerinde zaman alıcı bir çok yazım tekniği mevcuttur. Kod yazan bir programcı için anlaşılması kolay kodlar ve kolay yönetilebilirlik oldukça önemlidir. Go dili, programcılarının en çok şikayet ettiği konuların sadeleştirildiği bir dil olarak işlev yürütmesi amaçlanmıştır. Google gibi büyük bir teknoloji şirketinin arkasında bulunması, bu dilin iyi bir yere gelebileceğinin göstergelerinden biridir.

Not

Bu makale 01-03 Kasım 2019 tarihleri arasında Kocaeli'de gerçekleştirilen Uluslararası Marmara Fen Bilimleri Kongresinde (IMASCON 2019) sözlü bildiri olarak sunulmuş ve yeniden yapılandırılmıştır.

Kaynakça

- [1] SUMMERFIELD M., Programming in Go: Creating Applications for the 21st Century, Addison-Wesley Professional, 2012

- [2] Donovan, A. A., Alan, Kernighan B., W., The Go Programming Language, 2009, New Jersey, United States Of America
- [3] SEGUIN K., Little Go Book, <https://www.openmymind.net/assets/go/go.pdf>, 2017
- [4] Pike R., The Go Programming Language, http://9p.io/sources/contrib/ericvh/go-plan9/doc/go_talk-20091030.pdf, 2009
- [5] Go Programming Language Documents, <https://godoc.org/>, 2019. [Online] (Erişim Tarihi: 10.10.2019)
- [6] Dependency and Package Management in GoLang Microservices Apps, <https://www.xenonstack.com/blog/updates/dependency-management-package-golang/>, 2017. [Online] (Erişim Tarihi: 14.10.2019)
- [7] My 5 favourite features of Go and how to use them, 2016. [Online] <https://making.pusher.com/my-5-favourite-features-of-go-and-how-to-use-them/>, (Erişim Tarihi: 18.09.2019)
- [8] What is Compiler, 2017, [Online] <https://en.wikipedia.org/wiki/Compiler>, (Erişim Tarihi: 20.09.2019)
- [9] How To Install Go 1.13 on Ubuntu 18.04 & 16.04 LTS, 2017. [Online] <https://tecadmin.net/install-go-on-ubuntu/>, (Erişim Tarihi : 20.09.2019)
- [10] Go By Example, 2018. [Online] <https://gobyexample.com/>, (Erişim Tarihi: 20.11.2019)
- [11] http://9p.io/sources/contrib/ericvh/go-plan9/doc/go_talk-20091030.pdf, (Erişim Tarihi : 12.10.2019)