やさしい Go言語入門

日向俊二◉著

CUTT カットシステム

■**サンプルファイルのダウンロードについて**

本書掲載のサンプルファイルは、下記 URL からダウンロードできます。

http://----------

・本書の内容についてのご意見、ご質問は、お名前、ご連絡先を明記のうえ、小社出版部宛文書（郵送または E-mail）でお送りください。
・電話によるお問い合わせはお受けできません。
・本書の解説範囲を越える内容のご質問や、本書の内容と無関係なご質問にはお答えできません。
・匿名のフリーメールアドレスからのお問い合わせには返信しかねます。

はじめに

Go 言語はシンプルなプログラミング言語です。しかし、YouTube のような大規模で実用的なサイトが go 言語で構築されていることからもわかるように、Go 言語は実践的で強力なプログラミング言語でもあります。また、Go 言語の技術者は他のプログラミング言語の技術者より高収入であることも知られています。

Go 言語は、完全にフリーかつオープンソースであり、Linux、macOS、Windows、Android、iOS など、現在の主要な OS に対応しています。Go 言語はコンパイラーを活用することや並行処理が言語レベルで備わっているために、これまでにないパフォーマンスを期待できます。たとえば、インタープリタ言語である Python では実現できないことを、Go 言語でならば容易に実現できる場合があります。また、豊富なライブラリを利用できるので、開発効率が高まります。

Go 言語は、比較的新しい言語です。そのため、これまでの他のプログラミング言語の良い点と弱点をよく検討して、現代の要求にふさわしい言語として作成されています。たとえば、プログラマーは低レベルのメモリ管理に煩わされることはありません。メモリ管理はランタイムライブラリに任せておける一方で、Python や Java などのインタープリタ言語の実行効率を大幅に上回るパフォーマンスを実現しています。さらに、同じ 1 つのソースコードで多様な実行環境に対応することもできます。

本書はこのような特徴を持つ Go 言語のプログラミングをやさしく解説する書籍です。特に Go 言語の重要な事項に焦点を当てて、簡潔なサンプルプログラムを豊富に使ってわかりやすく解説しています。コンピュータの基本的な操作や、ファイルやディレクトリ（フォルダー）とそれらの基本的な操作、テキストファイルの作成と保存などについて知っていれば、誰でも本書を使って容易に Go 言語のプログラミングを学ぶことができます。

本書を参考にして Go 言語を楽しみながらマスターしてください。

2020 年 3 月　著者しるす

本書の表記

xyz　斜体で表記された文字（列）は、そこに具体的な文字や数値が入ることを表します。

[ ]　書式の説明で [ ] で囲まれている場合は、その項目を省略可能であることを示します。

>　OS のコマンドプロンプトを表します。Linux などの UNIX 系 OS での実行例では $ で示すこともあります。

補足説明や知っておくと良い事柄です。

ご注意

- 本書の内容は本書執筆時の状態で記述しています。執筆時の Go 言語の最新のバージョンは 1.13 です。将来、Go 言語のバージョンが変わるなど、何らかの理由で記述と実際とが異なる結果となる可能性があります。
- 本書は Go 言語や Go 言語で使うことができるパッケージについてすべて完全に解説するものではありません。必要に応じて Go 言語のドキュメントなどを参照してください。
- 本書のサンプルは、プログラミングを理解するために掲載するものです。実用的なアプリとして提供するものではありませんので、ユーザーのエラーへの対処やその他の面で省略してあるところがあります。
- 掲載しているプログラムコードの断片は、そのままでは実行できません。コードを実行するときには本書の内容をよく読んで理解してから実行してください。

本書に関するお問い合わせについて

本書に関するお問い合わせは、sales@cutt.co.jp にメールでご連絡ください。

なお、お問い合わせ内容は本書に記述されている範囲に限らせていただきます。特定の環境や特定の目的に対するお問い合わせ等にはお答えできませんので、あらかじめご了承ください。

お問い合わせの際には下記事項を明記してください。

- 氏名
- 連絡先メールアドレス
- 書名
- 記載ページ
- 問い合わせ内容
- 実行環境

もくじ

第1章

はじめての Go 言語

ここでは Go 言語の単純なプログラムを作成して実行してみながら、Go 言語の概要を学びます。

ここで紹介する Go 言語の構成要素については後の章で詳しく説明するので、この章では詳細はありのままに受け入れて、Go 言語でプログラムを作って実行するための作業の流れを掴んでください。

1.1 hello プログラム

伝統的に、プログラミング言語を学ぶときの最初の課題は、「hello, world」と表示するプログラムを作ることでした。ここでは Go 言語で「hello, world」と表示するプログラムを作って実行する方法を説明します。

◆ はじめてのプログラム

Go 言語で「hello, world」と表示するプログラムを作る方法はいろいろありますが、ここでは 1 行の文字列を出力する Go 言語の命令である `Println()` と呼ぶものを使って作ることにします。

次のソースプログラム（ソースリスト）をテキストエディターなどで入力してください。そして、`hello.go` という名前でファイルに保存します（各行の意味は後で説明します）。

リスト1.1●hello.go

```
// hello.go
package main

import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

プログラムは UTF-8 という文字コードで保存してください。

◆ プログラムのビルドと実行

このプログラムを実行するためには、ソースファイル hello.go に対してビルド（コンパイルともいう）と呼ぶ作業を行う必要があります。そのためには、このファイルを保存したディレクトリで次のコマンドを実行します。

```
go build [ソースファイル名]
```

次の例は Windows でファイルを「C:¥golang¥ch01¥hello」に保存したときの例です。

```
C:¥user>cd C:¥golang¥ch01¥hello
C:¥golang¥ch01¥hello>go build hello.go
```

これで実行可能ファイル hello.exe が生成されます。

また、ファイルを保存したディレクトリでファイル名を省略して「go build」というコマンドを実行することでもビルドできます（より複雑なプログラムでは省略するとビルドできない場合もあります）。

次の例は Windows でファイルを「C:¥golang¥ch01¥hello」に保存したときの例です。

```
C:¥user>cd C:¥golang¥ch01¥hello
C:¥golang¥ch01¥hello>go build
```

これで実行可能ファイル hello.exe が生成されます（go build コマンドを実行すると、実行可能ファイル名がそのソースファイルがあるディレクトリ名になります）。

「go build」でビルド（コンパイル）したときには、実行可能ファイル名はそのファイルがあるサブディレクトリ名と同じ名前になります。たとえば Windows で「C:¥golang¥ch01¥hello」にあるソースファイル hello.go をコンパイルしたら hello.exe になりますが、「C:¥work¥sample」にあるソースファイル hello.go をコンパイルしたら sample.exe になります。

hello.exe を実行するためには単に「hello」と入力します。

```
C:¥golang¥ch01¥hello>hello
hello, world
```

ここで出力された「hello, world」がこのプログラムの実行結果です。

Linux の場合も同様にします。次の例はファイルを「~/golang/ch01/hello」に保存したときの例です（「~/」はユーザーのホームディレクトリを表します）。

```
$ cd ~/golang/ch01/hello
$ go build
```

これで実行可能ファイル hello が生成されます。

生成された実行可能ファイル hello を実行するためには単に「./hello」と入力します。

```
$ ./hello
hello, world
```

ここで出力された「hello, world」がこのプログラムの実行結果です。

Note

Go 言語でビルドとは、ソースプログラムから実行可能なファイルを生成することです。この作業を（広義の）コンパイルと呼ぶこともあり、Go 言語のプログラムはソースプログラムをコンパイルして初めて実行できるので、コンパイル言語あるいはコンパイラー言語と呼ぶことがあります。一方、Python のようにビルドする必要はなく、ソースファイルをそのまま読み込んで実行する言語をインタープリタ言語といいます。インタープリタ言語はコンパイルが不要なので手軽なように見えますが、実行に時間がかかるという大きな欠点があります。Go 言語の実行可能ファイルは非常に高速で動作するので、特にアクセスが非常に多いサーバープログラムのようなプログラムで個々のリクエストに対して並列実行することで威力を発揮できます。

◆ ビルド実行コマンド ◆

ソースファイルをビルド（コンパイル）してその場で実行するためのコマンドとして、go run コマンドがあります。

hello.go をビルドして即実行するときには次のようにします。

```
> go run hello.go
```

この go run コマンドでプログラムを実行した場合は、実行可能ファイル（この場合は hello.exe や hello）は作成されません（go run コマンドでプログラムを実行する場合はソースファイル名を省略しません）。

1.2 hello プログラムの意味

ここでは hello プログラムの各行を詳しく説明します。

◆ プログラムの解説

先ほど入力したプログラムをもう一度見てみましょう。

リスト1.2●hello.go

```
// hello.go
package main

import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

1 行目の「// hello.go」はコメントです。Go 言語では「//」で始まって行末までがコメントです。

コメントはプログラムの実行に影響を与えません。この例では、プログラムファイル名をコメントにしています。なお、「//」で始まって行末までのコメントの他に、「/*」で始まり「*/」で終わるコメントを使うこともできます。この /* */ の形式のコメントは複数行に渡っても構いません。たとえば次のように記述できます。

```
/*
 * hello.go - 最初のサンプルプログラム
 */
```

2 行目の「package main」は、以下の内容が main というパッケージと呼ぶものに含まれることを示しています。今の段階では、これは実行するプログラムに必要なおまじないであると考えておいてください。

空行の次の行の「import "fmt"」は、fmt と呼ぶパッケージを使うことを宣言しています。この fmt は後でもう一度出てきます。

「func main() {」は、これが main という名前の関数であることを表しています。Go のアプリケーションプログラムは、main という名前の関数にある最初の文から実行される決まりになっています。この行の最後の「{」は main という名前の関数のブロックの最初であることを表しています。

関数とは、何らかの機能を持っていて、それを使う（呼び出すという）と何らかの作用をするものです。main という名前の関数は、このプログラムの機能を備えます。後で出てくる Println() という関数は、引数の値を出力するという機能を持っています。関数であることを表すために main() とか Println() というように名前の後に () を付けることがよくあります。関数については第 6 章「関数」で詳しく説明します。

7 行目の「fmt.Println("hello, world")」は実際に実行される唯一の文で、fmt という名前のパッケージの中にある Println() という関数を使って「hello, world」という文字列を出力して改行するための命令です。

fmt という名前のパッケージの中にある Println() という関数を表すために、fmt.Println() という表現を使います。

ここでは Println() の引数は "hello, world" だけですが、Println() の引数には任意の数の任意の型の値を指定できます。複数の引数を指定するときには「,」(カンマ) で区切ります。たとえば次のようにします。

```
fmt.Println("hello", 10, 'x')        // 「hello 10 120」と出力される
```

'x' が 120 と出力されるのは、文字コードの値が出力されるからです（10 進数で 120 は 16 進数で 78 であり、これは x という文字の文字コードを表します）。

「fmt.Println("hello, world")」という行の前のほうの空白は、行を右にずらして表現することでこの文が「{」と「}」で囲まれたブロックの内部にあることを見やすくするためのものです。このように、空白を入れて行を右にずらして見やすくすることをインデントといいます。インデントもプログラムの実行には影響を与えません。そのため、次のように書いても間違いではありません。

```
func main() {
fmt.Println("hello, world")
}
```

しかし、目で見てわかりにくいので、ブロックの内側のコードはインデントするべきです。

最後の行の「}」は、「func main() {」という行の行末にあった「{」に対応するもので、この場合は main という名前の関数のブロックの最後であることを表しています。

Go 言語では、インデントの幅は水平タブ 1 個分のスペースにすることが推奨されています。一般的には、水平タブ 1 個分のスペースは空白 8 個分（システムで変更可能）ですから、厳密には、「fmt.Println("hello, world")」の行は次のように記載するべきかもしれません。

```
func main() {
        fmt.Println("hello, world")
}
```

しかし、本書は書籍であるために表示できる幅が限られているという事情から、印刷してあるソースコードでは、インデントを空白 4 個分にしています。gofmt というツール（→付録 B.2）を使うと、インデントを決められた通りにすることができます。

1.3 Go 言語の日本語対応

Go 言語ではソースコードに Unicode 文字を使うので、日本語を含めたさまざまな文字を自由に使うことができます。

◆ 日本語のプログラム

Go 言語のプログラムで日本語を表示するのは簡単です。たとえば、次のようにすることで日本語を表示したり入力することができます。

リスト1.3●hellojp.go

```
// hellojp.go
package main

import "fmt"

func main() {
    var name string
    fmt.Printf("名前を入力してください：")
    fmt.Scan(&name)
    fmt.Printf("こんにちは、%sさん!", name)
}
```

入力された文字列を受け取る fmt.Scan() や、書式を指定して出力する fmt.Printf() については第３章で説明します。ここでは日本語文字列を使うことができるプログラム例としてとらえてください。

これをコンパイルして実行すると次のように「こんにちは、」に続けて入力された名前が表示されます。

```
>go build

>hellojp.exe
名前を入力してください：山本太郎
こんにちは、山本太郎さん!
```

関数や変数の名前に日本語を使うこともできます。

たとえば、次のプログラムのように、変数名や関数名を日本語にしてもビルドして実行することができます（変数や関数については第 2 章以降で説明します。ここではさまざまな名前に日本語を使えるという点に注目してください）。

リスト1.4●jpname.go

```
// jpname.go
package main

import "fmt"

func 名前表示(氏名 string) {
    fmt.Printf("こんにちは、%sさん!", 氏名)
}

func main() {
    var 名前 string
    fmt.Printf("名前を入力してください：")
    fmt.Scan(&名前)
    名前表示(名前)
}
```

しかし、一般的には名前を日本語にすることは国際化や可読性の点で好ましくないので、変数名や関数名は特に理由がない限り平易な英語を使うべきです。

なお、ここで使われている関数 fmt.Scan() や、変数とプログラマーが自分で定義する関数については後の章で説明しますので、ここでは名前に日本語も使える例として見るだけにとどめてください。