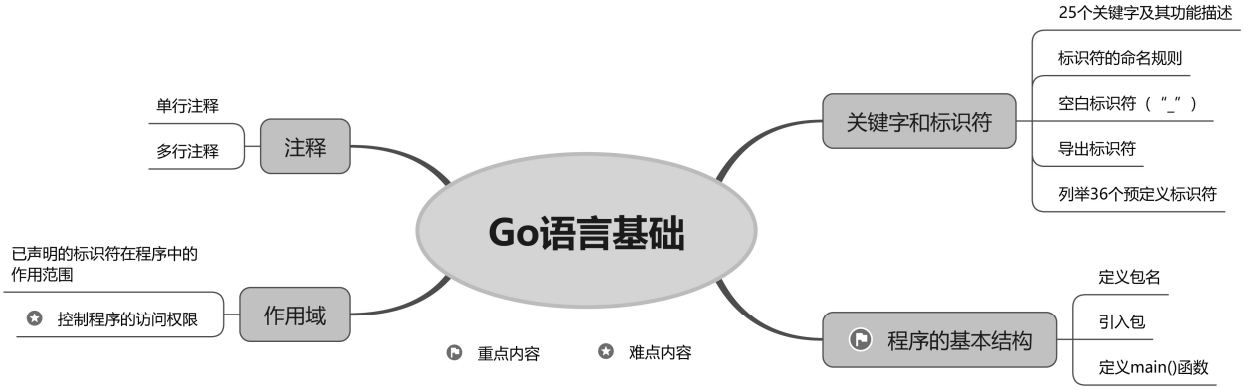
第 2 章

Go 语言基础

掌握并熟练应用 Go 语言需要充分理解 Go 语言的基础知识。本章讲解 Go 语言的基础知识，读者需要仔细阅读本章内容，进而让自己在学习 Go 语言的过程中达到事半功倍的效果。

本章的知识架构及重难点如下。



2.1 关键字和标识符

Go 语言的词法元素包括 5 种，分别是关键字、标识符、操作符、分隔符和字面量，它们是组成 Go 语言代码和程序的基本单位。本节主要介绍 Go 语言中的关键字和标识符。

2.1.1 关键字

关键字也称保留字，是指在 Go 语言中有特定含义，成为 Go 语言语法中一部分的单词。Go 语言中的关键字一共有 25 个，如表 2.1 所示。

表 2.1 Go 语言的关键字

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

在程序中，每个关键字都有着不同的功能含义，具体说明如表 2.2 所示。

表 2.2　关键字及其功能描述

关　键　字	功 能 描 述
break	跳出整个循环
default	设置默认值，常与 switch 语句和 select 语句结合使用
func	定义函数或方法
interface	定义接口
select	选择结构语句，常与 case 和 default 结合使用
case	选择结构，常与 switch 语句和 select 语句结合使用
defer	延时执行语句
go	启动并发执行
map	定义集合
struct	定义结构体
chan	定义通道
else	判断条件，与 if 结合使用
goto	跳转语句
package	定义包的名称
switch	选择结构语句，常与 case 和 default 结合使用
const	定义常量
fallthrough	在 switch 语句的 case 中使用 fallthrough，当 case 匹配成功时，强制下一个 case 语句
if	判断条件，常与 else 结合使用
range	迭代切片、管道或集合的元素
type	自定义数据类型
continue	跳过本次循环，直接进入下一次循环
for	循环语句
import	导入包
return	设置函数的返回值
var	定义变量

Go 语言中保留的关键字比较少，主要是为了简化编译过程中的代码解析。和其他语言一样，关键字不能作为标识符，否则程序会提示语法错误。

2.1.2　标识符命名规则

标识符是用户在编程时定义的名字，用于为各种变量、常量、方法、函数等命名。标识符的命名规则如下。

- ☑ 由 26 个英文字母、数字 0～9、下画线组成，但不能以数字开头。
- ☑ Go 语言中的标识符严格区分大小写。例如，UserName 与 username 代表两个不同的标识符。
- ☑ 标识符不能包含空格和特殊字符。
- ☑ 不能以关键字作为标识符，如 switch、if、for 等。

合法的标识符定义如下。

```
username
num1
next_city
```

不合法的标识符定义如下。

```
6room                    //不能以数字开头
default                  //关键字不能作为标识符
$age                     //不能包含特殊字符
```

虽然标识符可以任意命名，但是在编程时，最好还是使用便于记忆且有意义的名称，以增加程序的可读性。标识符的命名标准如下。

- ☑ 标识符的命名要尽量简短且有意义。
- ☑ 使用有意义的单词作为标识符，如名字定义为 name，宽度定义为 width。
- ☑ 如果不能使用一个单词命名标识符，则可以采用驼峰命名法，如 myBook、getValue。或使用下画线“_”连接所有单词，如 next_score。

2.1.3 空白标识符

空白标识符是由下画线“_”表示的特殊标识符，对于不被使用且存在的变量，可以使用空白标识符表示。空白标识符通常用于以下 3 种场景。

- ☑ 导入某个包，不调用包的任何变量或函数，只执行包的初始化函数 init()。
- ☑ 函数有多个返回值，但只需要使用一个返回值，不使用的返回值使用空白标识符表示。
- ☑ 类型断言，判断某个类型是否实现接口，没有实现则报告编译错误。

在函数中使用空白标识符表示不需要使用的返回值，代码如下。

```
package main

import "fmt"

func myfunc() (string, string, int) {        //自定义函数，设置 3 个返回值
    name := "张三"
    sex := "男"
    age := 20
    return name, sex, age
}
func main() {
    name, _, _ := myfunc()                   //调用函数，只获取第一个返回值
    fmt.Printf("姓名：%s", name)
}
```

2.1.4 导出标识符

Go 语言中的标识符首字母可以大写，如果包中定义的标识符首字母大写，则表示它可以被外部调用，这样的标识符叫作导出标识符。下面通过一个示例介绍导出标识符的用法。

在 D:\GoProject 路径下，新建文件夹 src，并在 VS Code 中打开 src 文件夹，再在 src 文件夹下新

建文件夹 test；在 test 文件夹中创建 demoA.go 文件。在 demoA.go 文件中定义包 test，再分别定义变量 Name、Age 和 Myfunc()函数。代码如下。

```
package test

var Name = "张三"
var Age int = 20

func Myfunc() string { //自定义函数
	address := "吉林省长春市"
	return address
}
```

在 D:\GoProject\src 路径下，创建 demoB.go 文件。在 demoB.go 文件中导入自定义包 test，并分别调用包中定义的变量和函数。代码如下。

```
package main

import (
	"fmt"

	"test"
)

func main() {
	fmt.Printf("姓名：%s\n", test.Name)
	fmt.Printf("年龄：%d\n", test.Age)
	fmt.Printf("地址：%s\n", test.Myfunc())
}
```

在编写 demoB.go 文件的过程中，import 代码中的“test”会一直提示错误。为了消除这个错误需要对 Go 语言开发环境进行设置。虽然已经通过设置环境变量把项目文件夹存放在 GOPATH 目录（即 D:\GoProject）下（参照图 1.16），但是“set GO111MODULE=on”配置项（参照图 1.21）不让 Go 命令行在 GOPATH 目录下查找.go 文件。也就是说，本示例如果不修改“set GO111MODULE=on”配置项，那么 Go 命令行就找不到 demoB.go 文件。

为了让 Go 命令行在 GOPATH 目录下查找 demoB.go 文件，需要把“set GO111MODULE=on”配置项修改为“set GO111MODULE=off”。修改步骤为：打开命令提示符对话框，输入“go env -w GO111MODULE=off”，按 Enter 键。

在退出命令提示符对话框后，务必重启 VS Code。重新进入 VS Code 后就会发现 import 代码块中“test”处的错误提示消失了。运行 demoB.go 文件，结果如下。

```
姓名：张三
年龄：20
地址：吉林省长春市
```

由此可知，在导入某个包时，如果包中定义的标识符的首字母大写，则该标识符可以被外部调用。

说明

后面的章节将启用 Module 功能。为了不影响后面章节的学习，在得到本示例的运行结果后，建议在命令提示符对话框中输入“go env -w GO111MODULE=on”，把“set GO111MODULE=off”配置项再修改为“set GO111MODULE=on”。

2.1.5 预定义标识符

在 Go 语言中还有一些特殊的标识符，叫作预定义标识符。预定义标识符包含 Go 语言的基本数据类型和内置方法，这些预定义标识符不能作为普通标识符使用。Go 语言中的预定义标识符如表 2.3 所示。

表 2.3 预定义标识符

append	bool	byte	cap	close	complex
complex64	complex128	uint16	copy	false	float32
float64	imag	int	int8	uint32	int32
int64	iota	len	make	new	nil
panic	uint64	print	println	real	recover
true	uint	uint8	uintptr		

2.2 程序的基本结构

在 Go 语言中，源码文件以.go 作为文件扩展名。.go 文件存放在包（文件夹）中，一个包由一个或多个.go 文件组成。文件中代码的开始是用 package 声明的，表示该文件属于哪个包。下面是一段 Go 代码。

```
package main

import "fmt"

func main() {
    fmt.Printf("Hello Go")
}
```

通过上述代码分析 Go 程序的基本结构，具体如下。

（1）第一行代码 package main 定义了包名，必须在源文件中非注释的第一行指明该文件属于哪个包。package main 表示可独立执行的程序，每个 Go 应用程序都包含一个名为 main 的包。

（2）import "fmt"表示导入内置包 fmt，该包主要实现数据的标准化输出。

（3）func main()是程序开始执行的函数。main 函数是每个可执行程序必须执行的函数，在一般情况下，main 函数是在启动后第一个执行的函数（如果有 init()函数，则先执行 init()函数）。

在 VS Code 中运行上述代码，输出结果如下。

```
Hello Go
```

另外，还可以在命令提示符中运行程序。将上述代码保存为 demo.go 文件，打开命令提示符，进入保存 demo.go 文件的文件夹中，输入命令“go run demo.go”，按 Enter 键即可执行程序，结果如图 2.1 所示。

在命令提示符中运行“go build demo.go”命令，Go 语言将.go 文件打包成.exe 文件。在当前目录中生成一个与.go 文件同名的.exe 可执行文件，运行.exe 文件，即可输出结果，如图 2.2 所示。

图 2.1　运行结果 1

图 2.2　运行结果 2

2.3　作　用　域

作用域表示已声明标识符表示的常量、变量、函数或包在程序中的作用范围。在对一个标识符的引用进行编译时，将从内层到外层查找该标识符的声明，如果没有找到，则报告编译错误。如果内层和外层都存在该标识符的声明，则以内层声明为主，且内层声明会覆盖外层声明。

在函数中定义两个变量 x 和 y，代码如下。

```
func add() (int){
    x := 10
    y := 20
    return x + y
}
```

在通常情况下，add 函数以外的程序无法访问 x 和 y 变量，这两个变量的作用域只在 add()函数内部有效。

通过作用域可以控制程序的访问权限。例如，在 main()函数外部和内部分别声明 3 个同名的变量 num，代码如下。

```
var num = 10
func main()   {
    num := 20
    {
        num := 30
        fmt.Println(num)
    }
    fmt.Println(num)
}
```

运行上述代码，结果如下。

```
30
20
```

在上述代码中，输出 num 变量的值，首先查找当前代码块中声明的变量，如果当前代码块中没有声明这个变量，就继续向外面一层进行查找，如果找到最外一层还没有找到，则报告编译错误。

2.4 注　　释

注释的作用是对程序进行解释和说明。编译系统在编译代码时会自动忽略注释，因此注释对于程序的功能实现不起任何作用。在程序中适当添加注释可以提高程序的可读性。为程序添加注释可以起到以下两种作用。

（1）解释程序某些语句的作用和功能，使程序更易于理解，通常用于代码的解释说明。

（2）用注释屏蔽一部分代码，使编译系统暂时忽略被注释的代码，等需要时再取消注释，这些代码就会再次发挥作用，通常用于代码的调试。

Go 语言提供两种注释符号："//" 和 "/*…*/"。其中，"//" 用于单行注释，"/*…*/" 用于多行注释。

☑ 单行注释

单行注释也叫行注释，是最常见的注释形式，可以在代码中的任何地方使用以//开头的单行注释。单行注释的格式如下。

```
//单行注释
```

☑ 多行注释

多行注释也叫块注释，分为开始和结束两部分，在需要注释的内容前输入 "/*" 表示注释开始，在注释内容结束后输入 "*/" 表示注释结束。多行注释一般用于描述包的文档或注释代码片段。多行注释的格式如下。

```
/*
第一行注释
第二行注释
......
*/
```

说明

（1）在多段注释之间可以使用空行分隔加以区分。

（2）对于代码中的变量、常量、函数等最好也应当加上对应的注释，有利于后期代码维护。

2.5 要 点 回 顾

本章向读者讲解的是 Go 语言基础，包含 4 个内容，即关键字和标识符、程序的基本结构、作用域和注释。在学习本章的过程中，读者首先要掌握 Go 语言有哪些关键字和为各种变量、常量、方法、函数等命名的方法；其次，明确 Go 程序的基本结构；再次，在使用变量时要注意常量、变量、函数等的有效范围，避免出现编译错误。最后，为增加代码的可读性，使用注释对程序进行解释和说明。