判定类问题求解

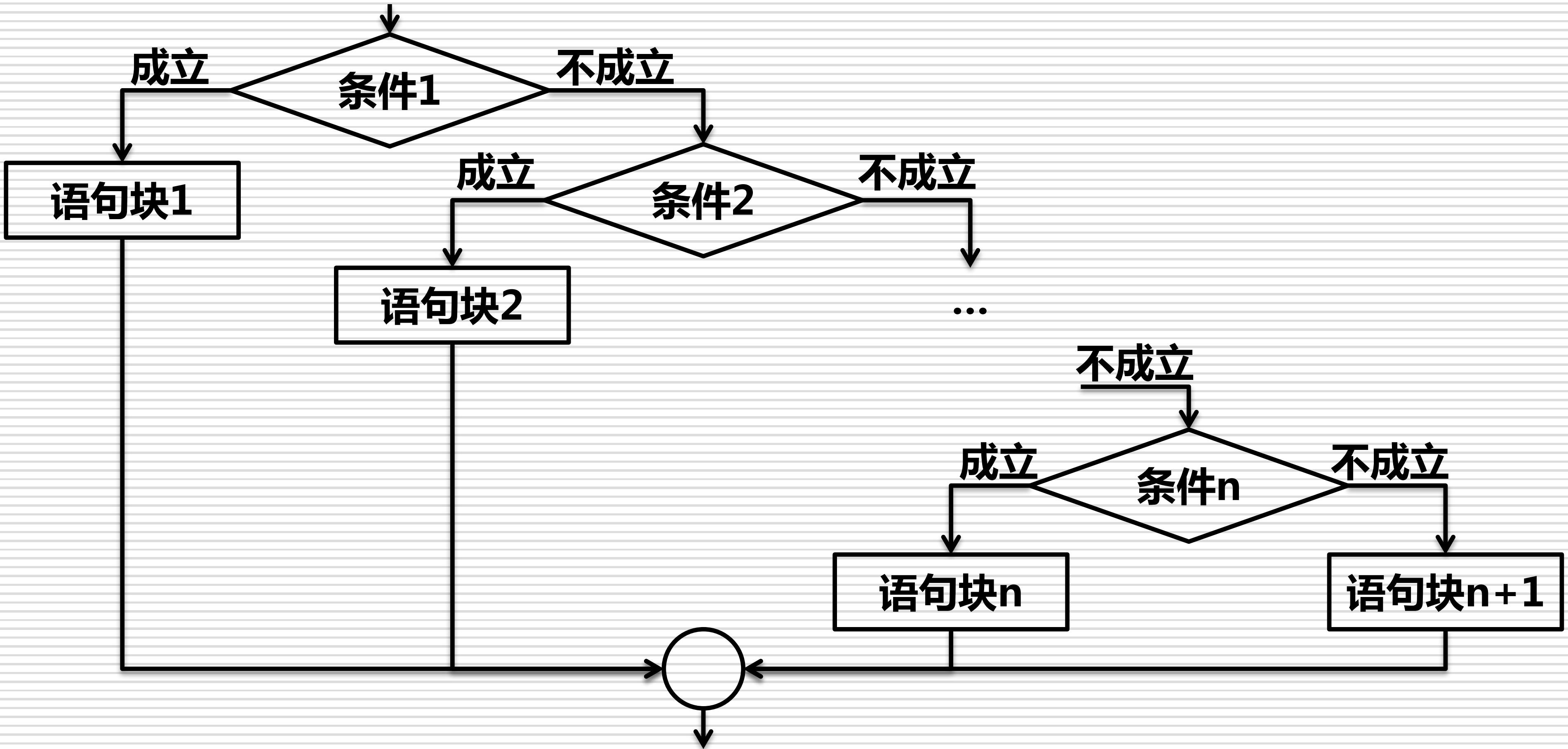
张 华

64174234@qq.com

内容

1.1什么是判定类问题

- 判定类问题的含义
  - 是指在求解过程中需要进行条件判断以确定执行哪些操作的问题，这类问题在技术层面上对应编程语言的选择结构。



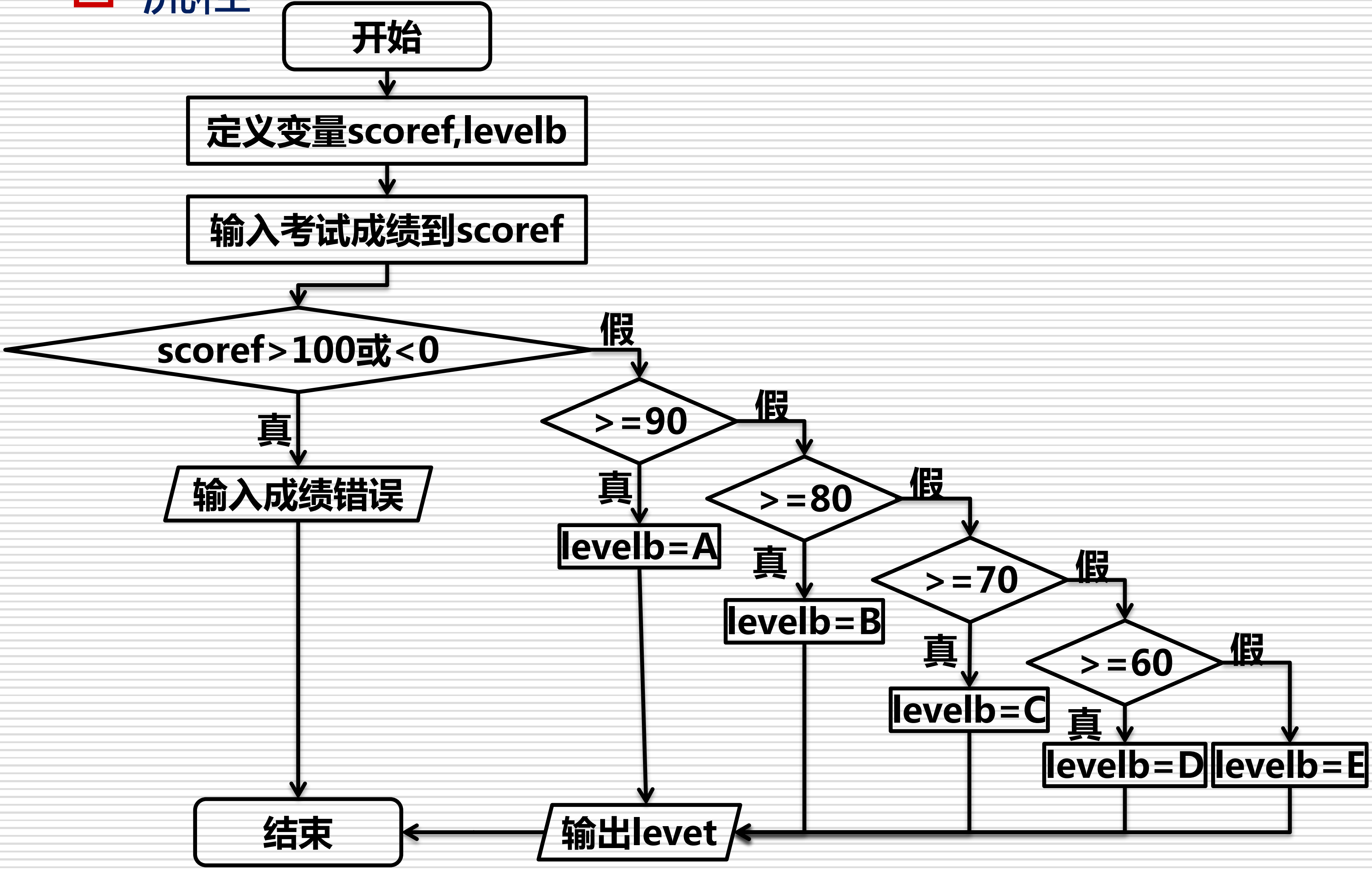
1.2将学生百分制成绩转换成ABCDE五个等级

- 输入学生的考试分数，90分以上(含90)为A等、60分以下(不含60)为E等，中间每隔10分划分为一个等级
- 量化：
  - 成绩变量scoref，float类型
  - 等级变量levelb，byte类型
- 抽象：
  - *leveln=f(scoref)*的关系如下

$$\text{leveln}=\begin{cases} A, & scoref \geq 90 \\ B, & 80 \leq scoref < 90 \\ C, & 70 \leq scoref < 80 \\ D, & 60 \leq scoref < 70 \\ E, & scoref < 60 \end{cases}$$

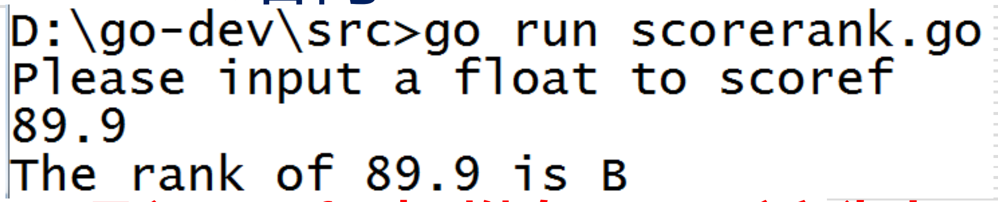
1.2将学生百分制成绩转换成ABCDE五个等级

- 流程

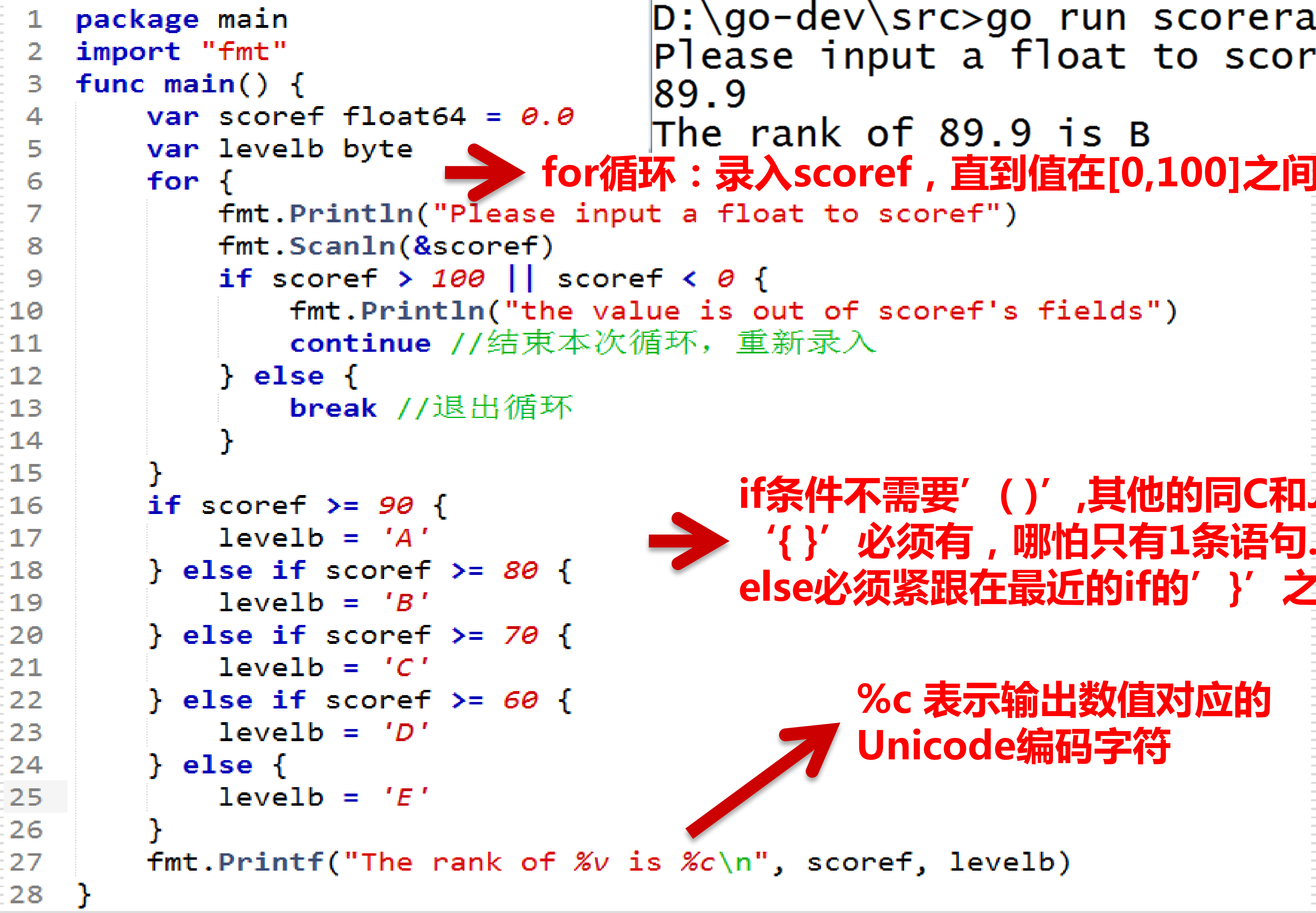


1.2将学生百分制成绩转换成ABCDE五个等级

- 实现——方式1：if-else结构



```
D:\go-dev\src>go run scorerank.go
Please input a float to scoref
89.9
The rank of 89.9 is B
```



```go
package main
import "fmt"
func main() {
    var scoref float64 = 0.0
    var levelb byte
    for {
        fmt.Println("Please input a float to scoref")
        fmt.Scanln(&scoref)
        if scoref > 100 || scoref < 0 {
            fmt.Println("the value is out of scoref's fields")
            continue //结束本次循环, 重新录入
        } else {
            break //退出循环
        }
    }
    if scoref >= 90 {
        levelb = 'A'
    } else if scoref >= 80 {
        levelb = 'B'
    } else if scoref >= 70 {
        levelb = 'C'
    } else if scoref >= 60 {
        levelb = 'D'
    } else {
        levelb = 'E'
    }
    fmt.Printf("The rank of %v is %c\n", scoref, levelb)
}
```

for循环：录入scoref，直到值在[0,100]之间为止

if条件不需要'( )',其他的同C和Java.
'{ }'必须有，哪怕只有1条语句.
else必须紧跟在最近的if的'}'之后.

%c 表示输出数值对应的Unicode编码字符

1.2将学生百分制成绩转换成ABCDE五个等级

实现——方式2：无条件switch结构

```go
package main
import "fmt"
func main() {
    var scoref float64 = 0.0
    var levelb byte
    for {
        fmt.Println("Please input a float to scoref")
        fmt.Scanln(&scoref)
        if scoref > 100 || scoref < 0 {
            fmt.Println("the value is out of scoref's fields")
            continue
        } else {
            break
        }
    }
    switch {
    case scoref <= 100 && scoref >= 90:
        levelb = 'A'
    case scoref < 90 && scoref >= 80:
        levelb = 'B'
    case scoref < 80 && scoref >= 70:
        levelb = 'C'
    case scoref < 70 && scoref >= 60:
        levelb = 'D'
    default:
        levelb = 'E'
    }
    fmt.Printf("The rank of %v is %c\n", scoref, levelb)
}
```

```
D:\go-dev\src>go run scoreranksw.go
Please input a float to scoref
88.5
The rank of 88.5 is B
```

没有条件的 switch 结构
这一构造使得可以用更清晰的形式来编写更长的 if-then-else 链。
无条件switch的结构如下

```
switch {
case condition1:
        ...
case condition2:
        ...
default:
        ...
}
```

1.2将学生百分制成绩转换成ABCDE五个等级

□ 实现——方式3：有条件swtich结构

```go
package main
import "fmt"
func main() {
	var scoref float64 = 0.0
	var levelb byte
	for {
		fmt.Println("Please input a float to scoref")
		fmt.Scanln(&scoref)
		if scoref > 100 || scoref < 0 {
			fmt.Println("the value is out of scoref's fields")
			continue
		} else {
			break
		}
	}
	switch int(scoref-50.0) / 10 { //以(scoref-50)的整数部分除以10作为分支条件
	case 4, 5:levelb = 'A'
	case 3:levelb = 'B'
	case 2:levelb = 'C'
	case 1:levelb = 'D'
	default:levelb = 'E'
	}
	fmt.Printf("The rank of %v is %c\n", scoref, levelb)
}
```

```
D:\go-dev\src>go run scorerankswcon.go
Please input a float to scoref
100
The rank of 100 is A

D:\go-dev\src>go run scorerankswcon.go
Please input a float to scoref
98
The rank of 98 is A
```

有条件switch，条件不局限于常量或整数
如果多个条件的执行分支相同，可用逗号间隔形成一条case分支

每个case都是从上至下逐一测试，直到匹配为止
当代码块只有一行时，可直接写到case后
每个分支执行完毕后，不需要break结束

1.2将学生百分制成绩转换成ABCDE五个等级

□ 实现——方式3：fallthrough的用法

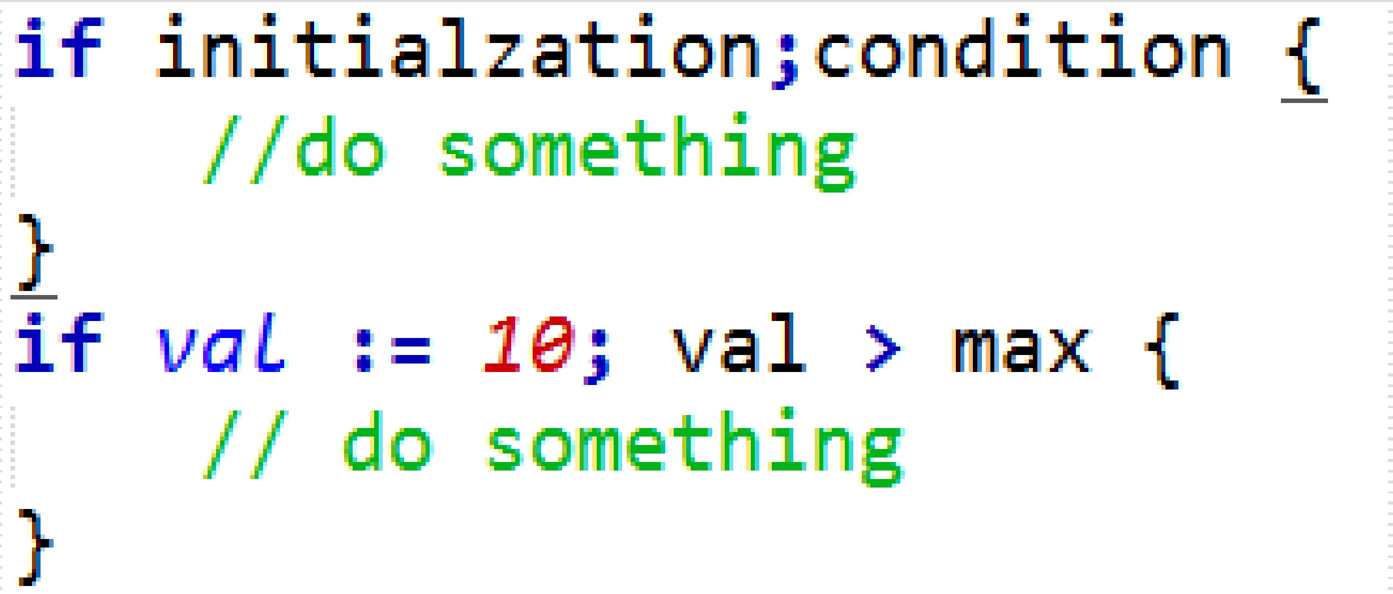
```
D:\go-dev\src>go run scorerankswcon.go
Please input a float to scoref
100
The rank of 100 is A

D:\go-dev\src>go run scorerankswcon.go
Please input a float to scoref
96.5
The rank of 96.5 is A
```

```go
package main
import "fmt"
func main() {
    var scoref float64 = 0.0
    var levelb byte
    for {
        fmt.Println("Please input a float to scoref")
        fmt.Scanln(&scoref)
        if scoref > 100 || scoref < 0 {
            fmt.Println("the value is out of scoref's fields")
            continue
        } else {
            break
        }
    }
    switch int(scoref-50.0) / 10 { //以(scoref-50)的整数部分除以10作为分支条件
    case 5: fallthrough
    case 4: levelb = 'A'
    case 3: levelb = 'B'
    case 2: levelb = 'C'
    case 1: levelb = 'D'
    default:    levelb = 'E'
    }
    fmt.Printf("The rank of %v is %c\n", scoref, levelb)
}
```

如果在执行完每个分支的代码后，还希望继续执行后续分支的代码，可以使用 fallthrough 关键字来达到目的
case 5满足后，就会执行case4的分支代码

1.3总结

- 判定类问题的含义(理解)
- 求解判定类问题的技术(掌握)
- switch的灵活使用方式(掌握)
- if 可以包含一个初始化语句（如：给一个变量赋值）。这种写法具有固定的格式（在初始化语句后方必须加上分号）：



```
if initialzation;condition {
    //do something
}
if val := 10; val > max {
    // do something
}
```

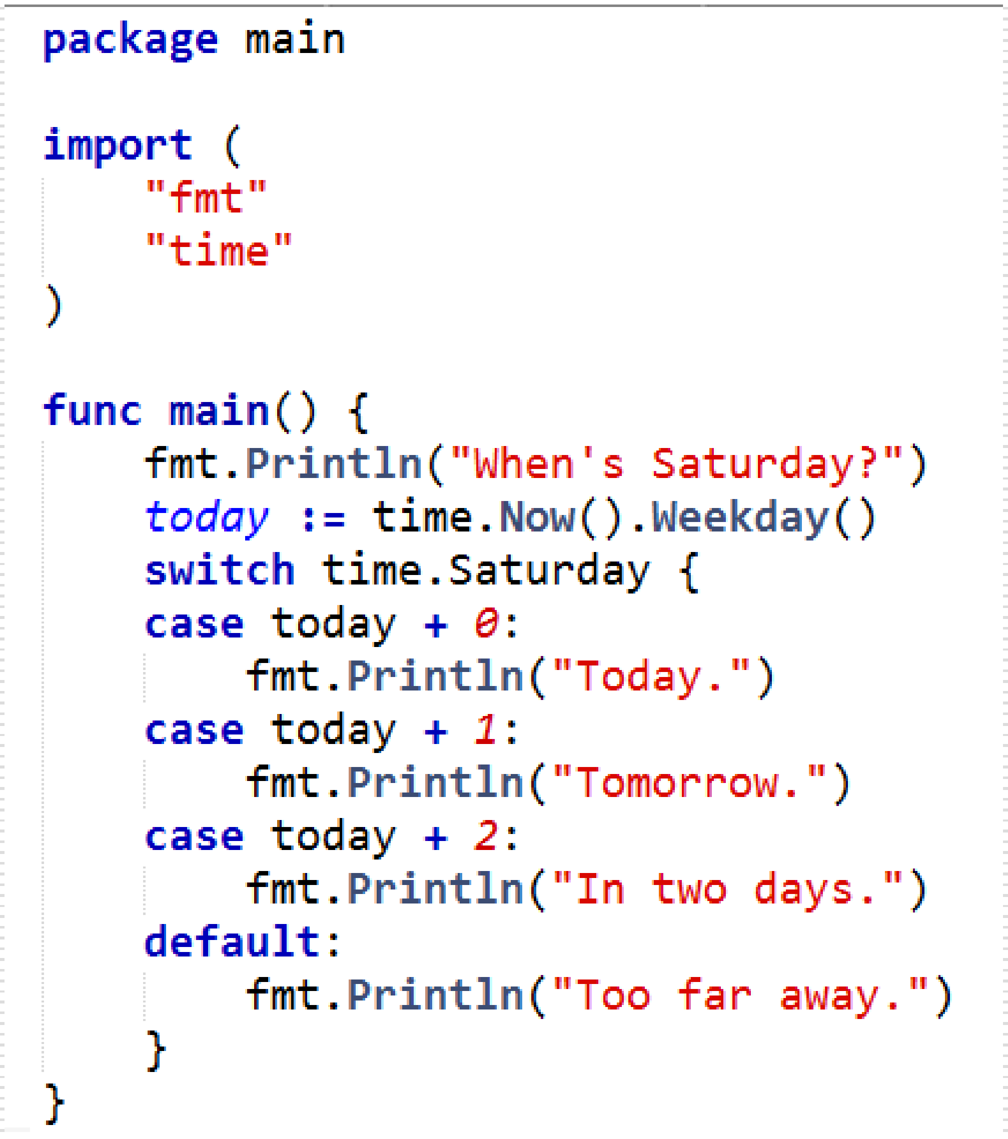
1.4思考(续)

- 请写出下面这段代码的结果

```go
package main

import "fmt"

func main() {
    var first int = 10
    var cond int

    if first <= 0 {

        fmt.Printf("first is less than or equal to 0\n")
    } else if first > 0 && first < 5 {

        fmt.Printf("first is between 0 and 5\n")
    } else {

        fmt.Printf("first is 5 or greater\n")
    }
    if cond = 5; cond > 10 {

        fmt.Printf("cond is greater than 10\n")
    } else {

        fmt.Printf("cond is not greater than 10\n")
    }
}
```

1.4思考(续)

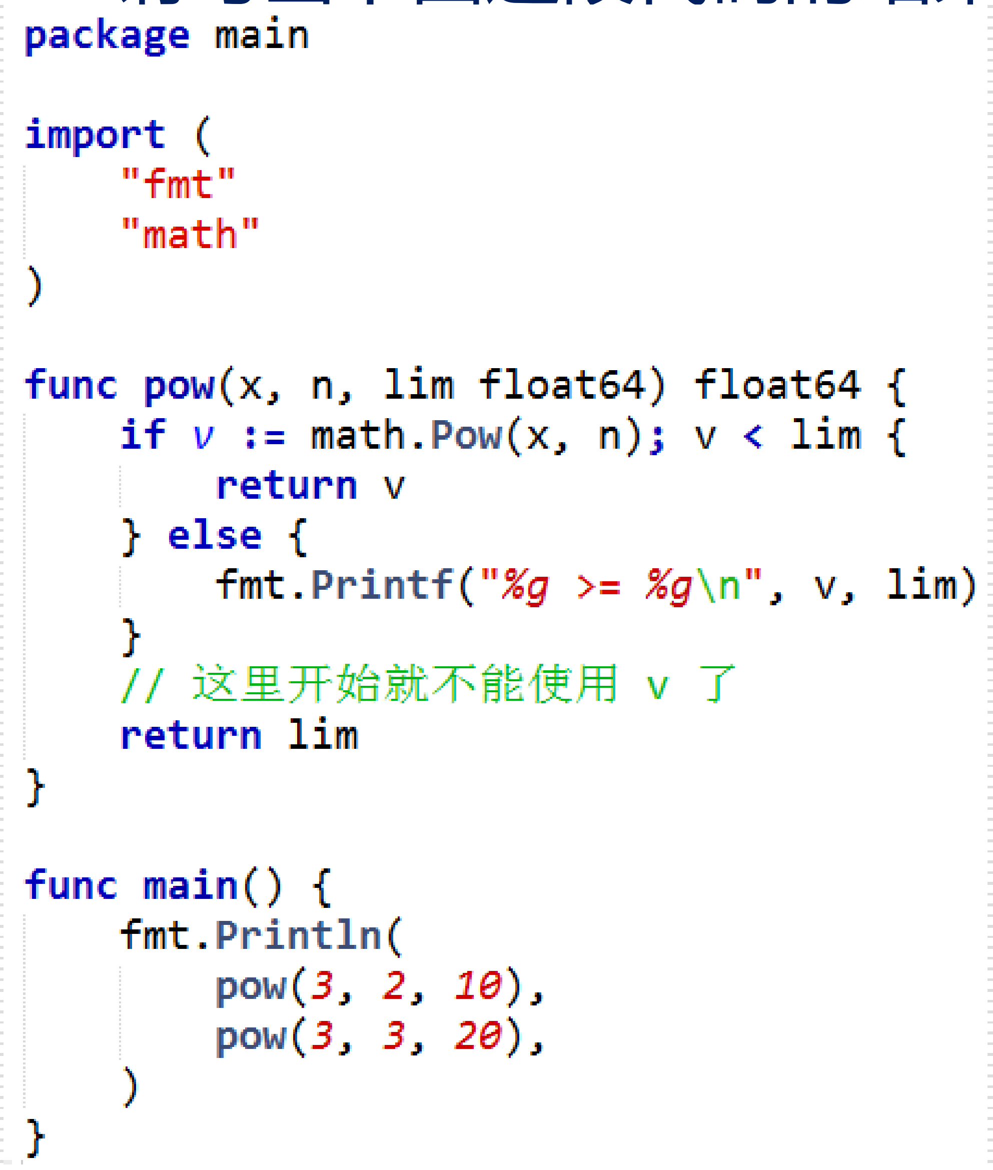
- 请说出下面这段代码的功能



```go
package main

import (
	"fmt"
	"time"
)

func main() {
	fmt.Println("When's Saturday?")
	today := time.Now().Weekday()
	switch time.Saturday {
	case today + 0:
		fmt.Println("Today.")
	case today + 1:
		fmt.Println("Tomorrow.")
	case today + 2:
		fmt.Println("In two days.")
	default:
		fmt.Println("Too far away.")
	}
}
```

1.4思考(续)

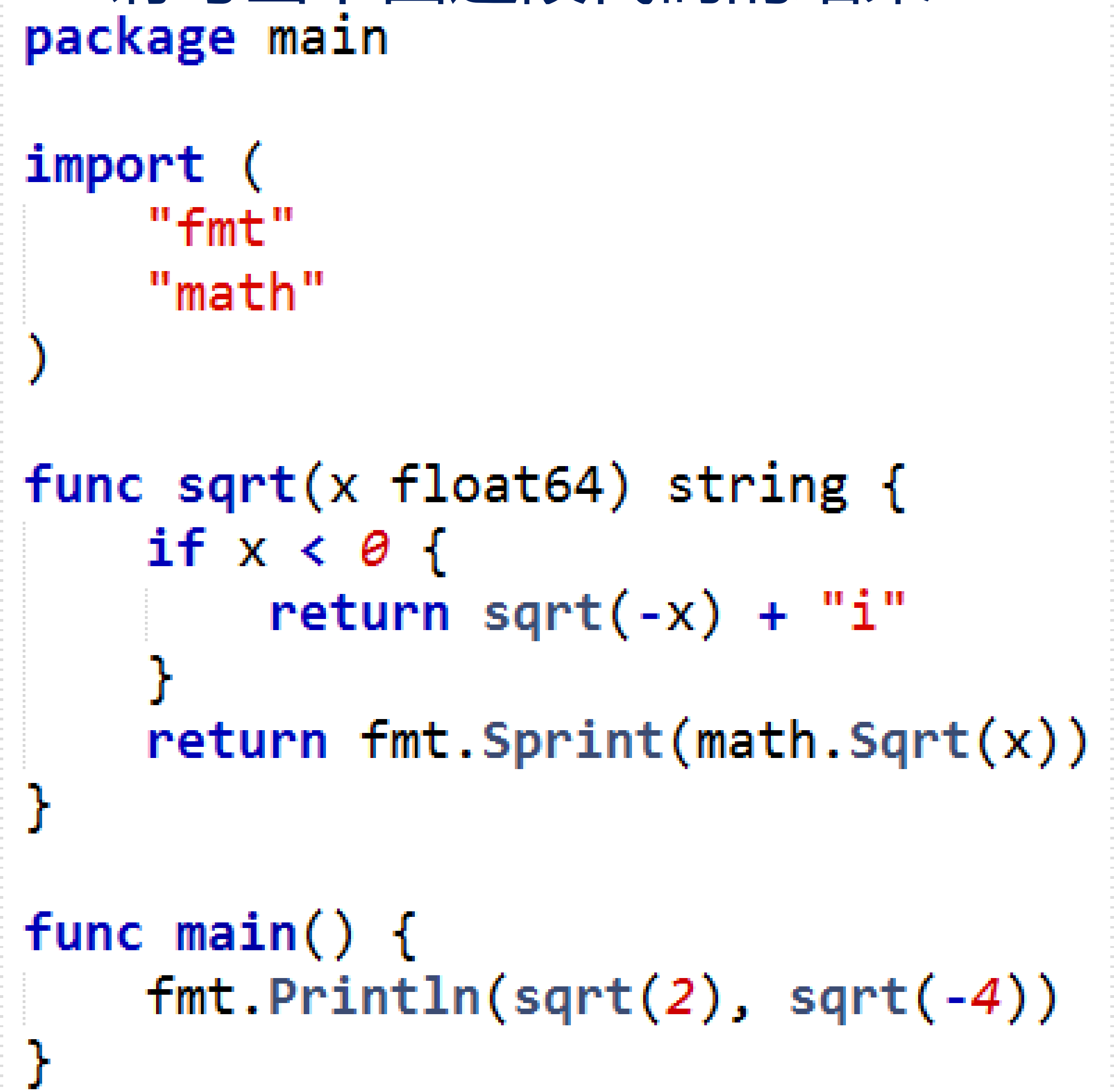
- 请写出下面这段代码的结果



```go
package main

import (
	"fmt"
	"math"
)

func pow(x, n, lim float64) float64 {
	if v := math.Pow(x, n); v < lim {
		return v
	} else {
		fmt.Printf("%g >= %g\n", v, lim)
	}
	// 这里开始就不能使用 v 了
	return lim
}

func main() {
	fmt.Println(
		pow(3, 2, 10),
		pow(3, 3, 20),
	)
}
```

1.4思考(续)

- 请写出下面这段代码的结果



```go
package main

import (
	"fmt"
	"math"
)

func sqrt(x float64) string {
	if x < 0 {
		return sqrt(-x) + "i"
	}
	return fmt.Sprint(math.Sqrt(x))
}

func main() {
	fmt.Println(sqrt(2), sqrt(-4))
}
```

1.4 思考

- 写一个 Season 函数，要求接受一个代表月份的数字，然后返回这个月份所在季节的名称（不用考虑月份的日期）。

Thank you very much

Any comments and suggestions are beyond welcome