time包、math包与随机数及键盘输入

目录：

一、time包

（一）、概述：

1、需要先import "time"。time包提供了时间的显示和测量用的函数。日历的计算采用的是公历。

（二）、time包中类型及方法

1、type Weekday

- func (d Weekday) String() string

2、type Month

- func (m Month) String() string

3、type Location

- func LoadLocation(name string) (*Location, error)
- func FixedZone(name string, offset int) *Location
- func (l *Location) String() string

4、type Time

- func Date(year int, month Month, day, hour, min, sec, nsec int, loc *Location) Time
- func Parse(layout, value string) (Time, error)
- func ParseInLocation(layout, value string, loc *Location) (Time, error)
- func Now() Time
- func Unix(sec int64, nsec int64) Time

- func (t Time) Location() *Location
- func (t Time) Zone() (name string, offset int)
- func (t Time) IsZero() bool
- func (t Time) Local() Time
- func (t Time) UTC() Time
- func (t Time) In(loc *Location) Time
- func (t Time) Unix() int64
- func (t Time) UnixNano() int64
- func (t Time) Equal(u Time) bool
- func (t Time) Before(u Time) bool
- func (t Time) After(u Time) bool
- func (t Time) Date() (year int, month Month, day int)
- func (t Time) Clock() (hour, min, sec int)
- func (t Time) Year() int
- func (t Time) Month() Month
- func (t Time) ISOWeek() (year, week int)
- func (t Time) YearDay() int
- func (t Time) Day() int
- func (t Time) Weekday() Weekday
- func (t Time) Hour() int
- func (t Time) Minute() int
- func (t Time) Second() int
- func (t Time) Nanosecond() int
- func (t Time) Add(d Duration) Time
- func (t Time) AddDate(years int, months int, days int) Time
- func (t Time) Sub(u Time) Duration
- func (t Time) Round(d Duration) Time
- func (t Time) Truncate(d Duration) Time
- func (t Time) Format(layout string) string
- func (t Time) String() string
- func (t Time) GobEncode() ([]byte, error)
- func (t *Time) GobDecode(data []byte) error
- func (t Time) MarshalBinary() ([]byte, error)
- func (t *Time) UnmarshalBinary(data []byte) error

- func (t Time) MarshalJSON() ([]byte, error)
- func (t *Time) UnmarshalJSON(data []byte) error
- func (t Time) MarshalText() ([]byte, error)
- func (t *Time) UnmarshalText(data []byte) error

5、type Duration

- func ParseDuration(s string) (Duration, error)
- func Since(t Time) Duration
- func (d Duration) Hours() float64
- func (d Duration) Minutes() float64
- func (d Duration) Seconds() float64
- func (d Duration) Nanoseconds() int64
- func (d Duration) String() string

6、type Timer

- func NewTimer(d Duration) *Timer
- func AfterFunc(d Duration, f func()) *Timer
- func (t *Timer) Reset(d Duration) bool
- func (t *Timer) Stop() bool

7、type Ticker

- func NewTicker(d Duration) *Ticker
- func (t *Ticker) Stop()

8、const (

```
Nanosecond  Duration = 1    //纳秒ns
Microsecond         = 1000 * Nanosecond     //微妙us
Millisecond         = 1000 * Microsecond   //毫秒ms
Second              = 1000 * Millisecond   //秒s
Minute              = 60 * Second
Hour                = 60 * Minute
)
```

（三）、time包中核心方法介绍

1、func Now() Time

- Now返回当前本地时间。

2、func (t Time) Local() Time

- Local将时间转成本地时区，但指向同一时间点的Time。

3、func (t Time) UTC() Time

- UTC将时间转成UTC和零时区，但指向同一时间点的Time。
- 国际上通过英国伦敦格林尼治天文台原址的那条经线称为0°经线，也叫本初子午线。

4、func Date(year int, month Month, day, hour, min, sec, nsec int, loc *Location) Time

- Date可以根据指定数值，返回一个时间。时区为loc，时间格式为：year-month-day hour:min:sec + nsec nanoseconds的时间点。loc可以是time.Local、time.UTC。string转time

5、func Parse(layout, value string) (Time, error)

- Parse能将一个格式化的时间字符串解析成它所代表的时间。就是string转time
- layout定义了参考时间：Mon Jan 2 15:04:05 -0700 MST 2006
- 如果缺少表示时区的信息，Parse会将时区设置为UTC。layout简写格式：Mon Jan 2 15:04:05 2006
- 预定义的ANSIC、UnixDate、RFC3339和其他版式描述了参考时间的标准或便捷表示。

6、func (t Time) Format(layout string) string

- Format根据layout指定的格式返回t代表的时间点的格式化文本表示。就是time转string
- layout定义了参考时间：Mon Jan 2 15:04:05 -0700 MST 2006

7、func (t Time) String() string

- String将时间格式化成字符串(time转string，相当于是固定格式的Format方法)，格式为："2006-01-02 15:04:05.999999999 -0700 MST"

8、func (t Time) Unix() int64

- Unix将t表示为Unix时间（时间戳，int64），即从时间点January 1, 1970

UTC到时间点t所经过的时间（单位秒）。

9、func (t Time) UnixNano() int64

- UnixNano将t表示为Unix时间，即从时间点January 1, 1970 UTC到时间点t所经过的时间（单位纳秒）。

10、func (t Time) Equal(u Time) bool

- 判断两个时间是否相同，会考虑时区的影响，因此不同时区标准的时间也可以正确比较。本方法和用t==u不同，这种方法还会比较地点和时区信息。

11、func (t Time) Before(u Time) bool

- 如果t代表的时间点在u之前，返回真；否则返回假。

12、func (t Time) After(u Time) bool

- 如果t代表的时间点在u之后，返回真；否则返回假。

13、func (t Time) Date() (year int, month Month, day int)

- 返回时间点t对应的年、月、日。

14、func (t Time) Year() int

- 返回时间点t对应的年份。

15、func (t Time) Month() Month

- 返回时间点t对应那一年的第几月。

16、func (t Time) Day() int

- 返回时间点t对应那一月的第几日。

17、func (t Time) Weekday() Weekday

- 返回时间点t对应的那一周的周几。

18、func (t Time) Clock() (hour, min, sec int)

- 返回t对应的那一天的时、分、秒。

19、func (t Time) Hour() int

- 返回t对应的那一天的第几小时，范围[0, 23]。

20、func (t Time) Minute() int

- 返回t对应的那一小时的第几分种，范围[0, 59]。

21、func (t Time) Second() int

- 返回t对应的那一分钟的第几秒，范围[0, 59]。

22、func (t Time) Nanosecond() int

- 返回t对应的那一秒内的纳秒偏移量，范围[0, 999999999]。

23、func (t Time) Sub(u Time) Duration

- 返回一个时间段t-u。如果结果超出了Duration可以表示的最大值/最小值，将返回最大值/最小值。要获取时间点t-d（d为Duration），可以使用t.Add(-d)。

24、func (d Duration) Hours() float64

- Hours将时间段表示为float64类型的小时数。

25、func (d Duration) Minutes() float64

- Minutes将时间段表示为float64类型的分钟数。

26、func (d Duration) Seconds() float64

- Seconds将时间段表示为float64类型的秒数。

27、func (d Duration) Nanoseconds() int64

- Nanoseconds将时间段表示为int64类型的纳秒数，等价于int64(d)。

28、func (d Duration) String() string

- 返回时间段采用"72h3m0.5s"格式的字符串表示。最前面可以有符号，数字+单位为一个单元，开始部分的0值单元会被省略；如果时间段<1s，会使用"ms"、"us"、"ns"来保证第一个单元的数字不是0；如果时间段为0，会返回"0"。

29、func ParseDuration(s string) (Duration, error)

- ParseDuration解析一个时间段字符串。一个时间段字符串是一个序列，每个片段包含可选的正负号、十进制数、可选的小数部分和单位后缀，如"300ms"、"-1.5h"、"2h45m"。合法的单位有"ns"、"us"/"μs"、"ms"、"s"、"m"、"h"。

30、func (t Time) Add(d Duration) Time

- Add返回时间点t+d。

31、func (t Time) AddDate(years int, months int, days int) Time

- AddDate返回增加了给出的年份、月份和天数的时间点Time。例如，时间点January 1, 2011调用AddDate(-1, 2, 3)会返回March 4, 2010。
- AddDate会将结果规范化，类似Date函数的做法。因此，举个例子，给时间点October 31添加一个月，会生成时间点December 1。（从时间点November 31规范化而来）

二、math包

（一）、概述：

1、使用时需要import "math"，math包提供了基本的数学常数和数学函数。

（二）、math包中函数

- func NaN() float64
- func IsNaN(f float64) (is bool)
- func Inf(sign int) float64
- func IsInf(f float64, sign int) bool
- func Float32bits(f float32) uint32
- func Float32frombits(b uint32) float32
- func Float64bits(f float64) uint64
- func Float64frombits(b uint64) float64
- func Signbit(x float64) bool
- func Copysign(x, y float64) float64

- func Ceil(x float64) float64
- func Floor(x float64) float64
- func Trunc(x float64) float64
- func Modf(f float64) (int float64, frac float64)
- func Nextafter(x, y float64) (r float64)
- func Abs(x float64) float64
- func Max(x, y float64) float64
- func Min(x, y float64) float64
- func Dim(x, y float64) float64
- func Mod(x, y float64) float64
- func Remainder(x, y float64) float64
- func Sqrt(x float64) float64
- func Cbrt(x float64) float64
- func Hypot(p, q float64) float64
- func Sin(x float64) float64
- func Cos(x float64) float64
- func Tan(x float64) float64
- func Sincos(x float64) (sin, cos float64)
- func Asin(x float64) float64
- func Acos(x float64) float64
- func Atan(x float64) float64
- func Atan2(y, x float64) float64
- func Sinh(x float64) float64
- func Cosh(x float64) float64
- func Tanh(x float64) float64
- func Asinh(x float64) float64
- func Acosh(x float64) float64
- func Atanh(x float64) float64
- func Log(x float64) float64
- func Log1p(x float64) float64
- func Log2(x float64) float64
- func Log10(x float64) float64
- func Logb(x float64) float64
- func Ilogb(x float64) int

- func Frexp(f float64) (frac float64, exp int)
- func Ldexp(frac float64, exp int) float64
- func Exp(x float64) float64
- func Expm1(x float64) float64
- func Exp2(x float64) float64
- func Pow(x, y float64) float64
- func Pow10(e int) float64
- func Gamma(x float64) float64
- func Lgamma(x float64) (lgamma float64, sign int)
- func Erf(x float64) float64
- func Erfc(x float64) float64
- func J0(x float64) float64
- func J1(x float64) float64
- func Jn(n int, x float64) float64
- func Y0(x float64) float64
- func Y1(x float64) float64
- func Yn(n int, x float64) float64

（三）、math包中核心函数介绍

1、func IsNaN(f float64) (is bool)

- 报告f是否表示一个NaN（Not A Number）值。

2、func Ceil(x float64) float64

- 返回不小于x的最小整数（的浮点值）

3、func Floor(x float64) float64

- 返回不大于x的最小整数（的浮点值）

4、func Trunc(x float64) float64

- 返回x的整数部分（的浮点值）。

5、func Abs(x float64) float64

- 返回x的绝对值

6、func Max(x, y float64) float64

- 返回x和y中最大值

7、func Min(x, y float64) float64

- 返回x和y中最小值

8、func Dim(x, y float64) float64

- 函数返回x-y和0中的最大值

9、func Mod(x, y float64) float64

- 取余运算，可以理解为 x-Trunc(x/y)*y，结果的正负号和x相同

10、func Sqrt(x float64) float64

- 返回x的二次方根

11、func Cbrt(x float64) float64

- 返回x的三次方根，特例如下：

12、func Hypot(p, q float64) float64

- 返回Sqrt(p*p + q*q)

13、func Pow(x, y float64) float64

- 返回x^y

14、func Sin(x float64) float64

- 求正弦。

15、func Cos(x float64) float64

- 求余弦。

16、func Tan(x float64) float64

- 求正切。

17、func Log(x float64) float64

- 求自然对数

18、func Log2(x float64) float64
- 求2为底的对数。

19、func Log10(x float64) float64
求10为底的对数。

三、随机数(math/rand包)

（一）、概述：
1、使用时需要import "math/rand"，rand包实现了伪随机数生成器。
2、随机数从资源生成。包水平的函数都使用的默认的公共资源。该资源会在程序每次运行时都产生确定的序列。如果需要每次运行产生不同的序列，应使用Seed函数进行初始化。默认资源可以安全的用于多协程并发。

（二）、rand包中类型及方法
1、type Source
- func NewSource(seed int64) Source

2、type Rand
- func New(src Source) *Rand
- func (r *Rand) Seed(seed int64)
- func (r *Rand) Int() int
- func (r *Rand) Int31() int32
- func (r *Rand) Int63() int64
- func (r *Rand) Uint32() uint32
- func (r *Rand) Intn(n int) int
- func (r *Rand) Int31n(n int32) int32
- func (r *Rand) Int63n(n int64) int64
- func (r *Rand) Float32() float32
- func (r *Rand) Float64() float64
- func (r *Rand) NormFloat64() float64
- func (r *Rand) ExpFloat64() float64

- func (r *Rand) Perm(n int) []int

3、type Zipf

- func NewZipf(r *Rand, s float64, v float64, imax uint64) *Zipf
- func (z *Zipf) Uint64() uint64
- func Seed(seed int64)
- func Int() int
- func Int31() int32
- func Int63() int64
- func Uint32() uint32
- func Intn(n int) int
- func Int31n(n int32) int32
- func Int63n(n int64) int64
- func Float32() float32
- func Float64() float64
- func NormFloat64() float64
- func ExpFloat64() float64
- func Perm(n int) []int

（三）、rand包中核心方法介绍

1、func NewSource(seed int64) Source

使用给定的种子创建一个伪随机资源。

2、func New(src Source) *Rand

返回一个使用src生产的随机数来生成其他各种分布的随机数值的*Rand。

3、func (r *Rand) Seed(seed int64)

使用给定的seed来初始化生成器到一个确定的状态。

4、func (r *Rand) Int() int

返回一个非负的伪随机int值。

5、func (r *Rand) Intn(n int) int

返回一个取值范围在[0,n)的伪随机int值，如果n<=0会panic。

6、func (r *Rand) Float64() float64
返回一个取值范围在[0.0, 1.0)的伪随机float64值。

（四）、获取随机数的几种方式：
1、通过默认随机数种子获取随机数

- rand.Int()
- rand.Float64()
- rand.Intn(n) // 例如获取0-n之间随机数
- 总是生成固定的随机数。默认情况下，随机数种子都是1。seed是一个64位整数。

2、动态随机数种子生成随机资源，实例随机对象来获取随机数

```
s1 := rand.NewSource(time.Now().UnixNano())
r1 := rand.New(s1)
randnum := r1.Intn(n)  // 例如获取0-n之间随机数
```

3、简写形式：动态变化随机数种子来获取随机数
（1）、获取整型随机数[0,10]

- rand.Seed(time.Now().UnixNano())
- rand.Intn(10)

（2）、获取浮点型0.0至1.0之间的随机数

- rand.Seed(time.Now().UnixNano())
- rand.Float64()

（3）、获取两数之间随机数[m , n]

- rand.Seed(time.Now().UnixNano())
- 随机数 = **rand.Intn(n - m + 1) + m**
- 例如：获取[5,11]之间随机数： rand.Intn(7) + 5

四、键盘输入

（一）、scanln

1、fmt.scanln()

2、示例代码：

```
package main
import "fmt"
func main() {
  username := ""
  age := 0
  fmt.Scanln(&username, &age)
  fmt.Println("账号信息为：", username, age)
  fmt.Printf("用户名是：%q ，  年龄是：%d \n", username, age)
  fmt.Printf("用户名是：%s ，  年龄是：%d \n", username, age)
  fmt.Println(&username)
}
```

（二）、随机数+键盘输入案例——猜数字游戏

```
package main
import (
  "fmt"
  "math/rand"
  "strings"
  "time"
)

func main() {
  play()
}

func play() {
  target := generateRandNum(10, 100)
  //fmt.Println("产生随机数：", target)
  fmt.Println("请输入随机数：")
  fmt.Println(strings.Repeat("-" , target))
```

```
    //记录猜测的次数
    count := 0
    for {
        count++
        yourNum := 0
        fmt.Scanln(&yourNum)
        //fmt.Scanf("%d", &yourNum)

        if yourNum < target {
            fmt.Println("小了❌")
        } else if yourNum > target {
            fmt.Println("大了❌")
        } else {
            fmt.Println("正确✅")
            fmt.Printf("您一共猜测了 %d 次！\n", count)
            fmt.Println("-------------------------")
            play()
        }
        alertInfo(count, target)
    }
}

//生成随机数
func generateRandNum(min int, max int) int {
    rand.Seed(time.Now().UnixNano())
    return rand.Intn(max-min+1) + min
}

//提示信息
func alertInfo(count int, target int) {
    if count >= 6 {
        fmt.Printf("您一共猜测了 %d 次都没有猜中，太笨了！😓\n", count)
        fmt.Println("正确结果是：", target)
        fmt.Println("----------------------------")
```

```
        fmt.Println("")
        play()
    }
}
```