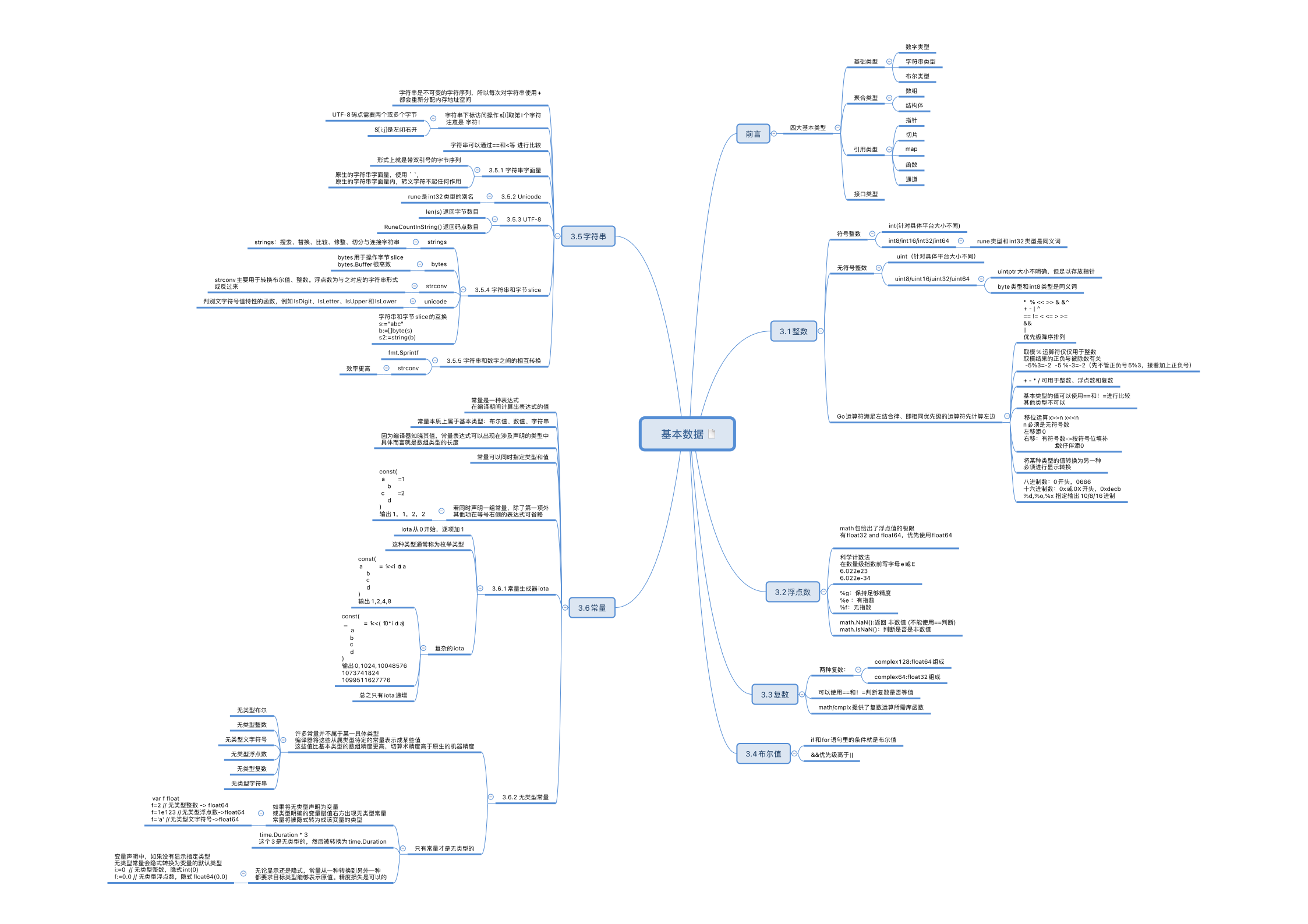
基本数据
前言
四大基本类型
基础类型
数字类型
字符串类型
布尔类型
聚合类型
数组
结构体
引用类型
指针
切片
map
函数
通道
接口类型
3.1整数
符号整数
int(针对具体平台大小不同)
int8/int16/int32/int64
rune类型和int32类型是同义词
无符号整数
uint（针对具体平台大小不同）
uint8/uint16/uint32/uint64
uintptr大小不明确，但足以存放指针
byte类型和int8类型是同义词
Go运算符满足左结合律、即相同优先级的运算符先计算左边
* % << >> & &^
+ - | ^
== != < <= > >=
&&
||
优先级降序排列
取模%运算符仅仅用于整数
取模结果的正负与被除数有关
-5%3=-2 -5 %-3=-2（先不管正负号5%3，接着加上正负号）
+ - * / 可用于整数、浮点数和复数
基本类型的值可以使用==和！=进行比较
其他类型不可以
移位运算 x>>n x<<n
n必须是无符号数
左移添0
右移：有符号数->按符号位填补
数仔伴添0
将某种类型的值转换为另一种
必须进行显示转换
八进制数：0开头，0666
十六进制数：0x或0X开头，0xdecb
%d,%o,%x 指定输出10/8/16进制
3.2浮点数
math包给出了浮点值的极限
有float32 and float64，优先使用float64
科学计数法
在数量级指数前写字母e或E
6.022e23
6.022e-34
%g：保持足够精度
%e：有指数
%f：无指数
math.NaN():返回 非数值（不能使用==判断）
math.IsNaN()：判断是否是非数值
3.3复数
两种复数：
complex128:float64组成
complex64:float32组成
可以使用==和！=判断复数是否等值
math/cmplx提供了复数运算所需库函数
3.4布尔值
if和for语句里的条件就是布尔值
&&优先级高于||
3.5字符串
字符串是不可变的字符序列，所以每次对字符串使用+都会重新分配内存地址空间
字符串下标访问操作s[i]取第i个字符
注意是 字符！
UTF-8码点需要两个或多个字节
S[i:j]是左闭右开
字符串可以通过==和<等 进行比较
3.5.1 字符串字面量
形式上就是带双引号的字节序列
原生的字符串字面量，使用``，
原生的字符串字面量内，转义字符不起任何作用
3.5.2 Unicode
rune是int32类型的别名
3.5.3 UTF-8
len(s)返回字节数目
RuneCountInString()返回码点数目
3.5.4 字符串和字节slice
strings
strings：搜索、替换、比较、修整、切分与连接字符串
bytes
bytes用于操作字节slice
bytes.Buffer很高效
strconv
strconv主要用于转换布尔值、整数。浮点数为与之对应的字符串形式
或反过来
unicode
判别文字符号值特性的函数，例如IsDigit、IsLetter、IsUpper和IsLower
字符串和字节slice的互换
s:="abc"
b:=[]byte(s)
s2:=string(b)
3.5.5 字符串和数字之间的相互转换
fmt.Sprintf
strconv
效率更高
3.6常量
常量是一种表达式
在编译期间计算出表达式的值
常量本质上属于基本类型：布尔值、数值、字符串
因为编译器知晓其值，常量表达式可以出现在涉及声明的类型中
具体而言就是数组类型的长度
常量可以同时指定类型和值
const(
a =1
b
c =2
d
)
输出1，1，2，2
若同时声明一组常量，除了第一项外
其他项在等号右侧的表达式可省略
3.6.1 常量生成器iota
iota从0开始，逐项加1
这种类型通常称为枚举类型
const(
a = 1<<i d a
b
c
d
)
输出1,2,4,8
复杂的iota
const(
_ = 1<<(10*i d a)
a
b
c
d
)
输出0,1024,10048576
1073741824
1099511627776
总之只有iota递增
3.6.2 无类型常量
许多常量并不属于某一具体类型
编译器将这些从属类型待定的常量表示成某些值
这些值比基本类型的数组精度更高，切算术精度高于原生的机器精度
无类型布尔
无类型整数
无类型文字符号
无类型浮点数
无类型复数
无类型字符串
只有常量才是无类型的
如果将无类型声明为变量
或类型明确的变量赋值右方出现无类型常量
常量将被隐式转为成该变量的类型
var f float
f=2 // 无类型整数 -> float64
f=1e123 //无类型浮点数->float64
f='a' //无类型文字符号->float64
time.Duration * 3
这个3是无类型的，然后被转换为time.Duration
无论显示还是隐式，常量从一种转换到另外一种
都要求目标类型能够表示原值。精度损失是可以的
变量声明中，如果没有显示指定类型
无类型常量会隐式转换为变量的默认类型
i:=0 // 无类型整数，隐式int(0)
f:=0.0 // 无类型浮点数，隐式float64(0.0)