中国科学院大学
University of Chinese Academy of Sciences

中国科学院计算技术研究所
Institute Of Computing Technology Chinese Academy Of Sciences

计算机科学导论实验

编程实验-信息隐藏

计算机科学导论实验课程助教组
中国科学院大学

2020年4月17日

内容大纲

- 课程说明
- 实验准备
- 实验示例
- 编程知识
- 控制流

实验示例

- 所有的 Golang 源代码的文件名均以 “.go” 为扩展名
- //hello_world.go

```
package main                    程序代码所在的包（包含函数和变量）
import “fmt”                    fmt 提供了 格式化输入和输出函数
func main() {                   声明 main 函数（对输入参数执行操作，输出计算结果）
    fmt.Println(“Hello World!”)     执行代码段
}
```

代码示例

```
package main
func main() {
}
```

(a)

```
package main
import "fmt"
func main() {
        fmt.Println("hello!")
}
```

(b)

```
package main
func main ( ) {
}
```

(c)

```
package main
func main { } (
)
```

(d)

```
package main
import {
        "fmt"
}
func main() {
        fmt.Println("hello!")
}
```

(e)

```
package main
func main() {
        fmt.Println("hello!")
}
```

(f)

内容大纲

- 课程说明
- 实验准备
- 实验示例
- 编程知识
- 控制流

编程知识

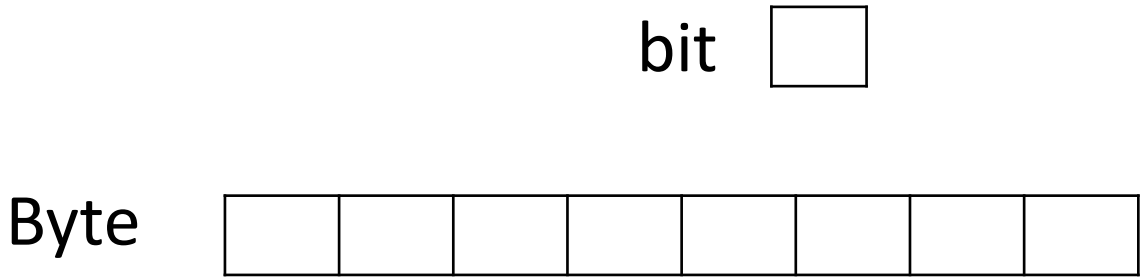
- 网站与教程
  - 官方网站：https://golang.org/
  - 在线教程：https://tour.golang.org/
  - 中文教程：https://www.runoob.com/go/go-tutorial.html

编程知识 —存储

- 比特（bit），存储值为0或1

- 字节（Byte），8个bit

- KB，2^{10} B
- MB，2^{10} KB
- GB，2^{10} MB
- TB，2^{10} GB
- PB，2^{10} TB



编程知识

- 数据类型和操作

类型		大小/B	取值	zero value
布尔值	bool	N/A	true/false	false
整数	byte	1	$[0,255]$	0
	int	8	$[-2^{63}, 2^{63}-1]$	0
	uint	8	$[0,2^{64}-1]$	0
浮点数	float64	8	使用IEEE-754 64位标准	0.0

编程知识 —变量

- 变量声明

```
var i int              // zero value: 0
var i,j,k int          // zero value: 0
var x float64          // zero value: 0.0
```

- 变量初始化

```
var i,j int = 1,2
var y byte = ‘a’       // one-byte integer: 97
var z byte = 97        // one-byte integer: 97
```

这里的’a’和97表示相同的意义，都会以其ASCII码的形式存储在计算机中

ASCII字符集

ASCII码=$D_6D_5D_4D_3D_2D_1D_0$

$D_3D_2D_1D_0$ \ $D_6D_5D_4$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

编程知识 —变量

- 变量赋值与类型转换

```
var i int = 1                    // 变量的初始化
var j int
j = i                            // 变量的赋值

var f float64 = i                // error
var f float64 = float64(i)       // 类型转换
var u uint = uint(f)             // 类型转换
```

编程知识 —变量

- 短变量声明

```
i := 10                                          // i: int
a , b := true , false                            // a , b : bool
fmt.Printf ("Type: %T Value: %v\n", i, i)        //打印变量的类型和值
```

- 常量

```
const Pi float64 = 3.14
const Pi = 3.14
```

```
const Pi          // error
Pi = 3.1415       // error
```

编程知识 —数组的声明和初始化

- 数组也是一种数据类型
- “名字 [n]T ”表示拥有 n 个 T 类型变量的数组，如

```
var a [10]int
var primes [6]int = [6]int {2,3,5,7,11,13}
var n int = 6
var a [n]int      // error
```

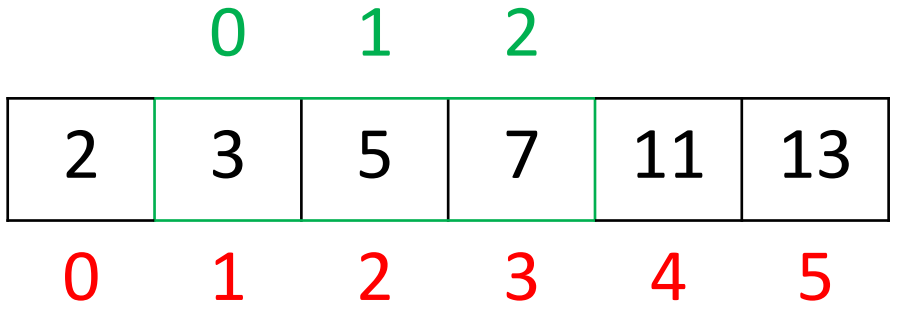
注：数组的长度 n 是其类型的一部分，因此数组不能改变大小【切片】

编程知识 —数组的访问

- 数组的下标从 0 开始。
  假设数组的长度为 n，则合法的下标为 $0,1,\ldots,n-1$

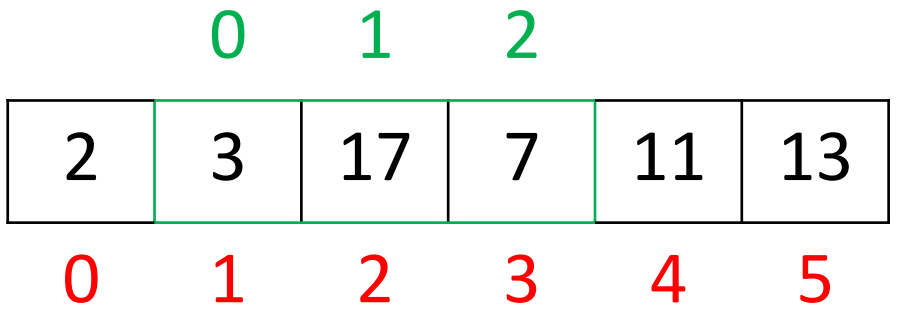
- 记 a 为某个数组的名称，则用 a[i] 表示数组的第 i 个元素，
  如 a[0] 表示数组的第 0 个元素，a[n] 是一个非法的表示

编程知识 —切片



- “名字 [ ]T” 表示一个元素类型为T的切片

```
var primes [6]int = [6]int {2,3,5,7,11,13}      // 数组
var s [ ]int = primes[1:4]                      // 切片
```



- 在令s[1]=17后，primes[2]=17

  - 切片并不存储任何数据，它只是描述了底层数组中的一段
  - 切片底层存储是数组，修改切片的某下标对应元素可以改变数组相应元素
  - [start: end]，从 start (包含) 到 end (不包含)；若省去 end，则表示到最后

编程知识 —切片与数组

1、通过数组来创建切片

- var a [3]bool = [3]bool {true, true, false} // a是一个数组
- var s [ ]bool = [ ]bool {true, true, false} // s 是一个切片
- 注：回顾一下，切片并不存储任何数据。因此，第二个声明首先会创建一个值为[true, true, false]的底层数组（无有名字），而后在其上创建切片s

2、直接初始化切片

- var s [ ]int = make([ ]int, 5)

3、通过 make 函数来产生切片

编程知识 —字符串

- var str1 string = “directly declaration”
- var byte_array [5]byte = [5]byte {72,101,108,108,111}
- var str2 string = string(byte_array[0:])

- 注：
  - 数组、切片、字符串都有长度，可用 len() 函数获取
  - 下标都是从 0 开始，故最后一个元素的下标是 “长度-1”

内容大纲

- 课程说明
- 实验准备
- 实验示例
- 编程知识
- 控制流

控制流

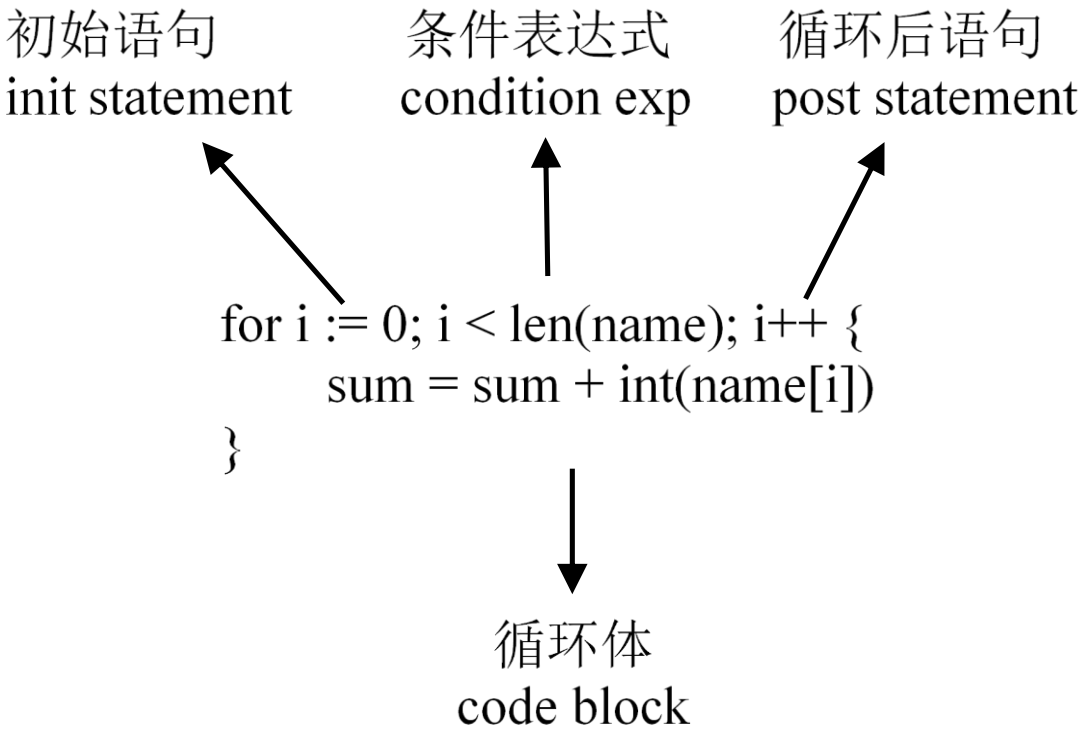
- 顺序流
  - 按照代码的编写顺序执行

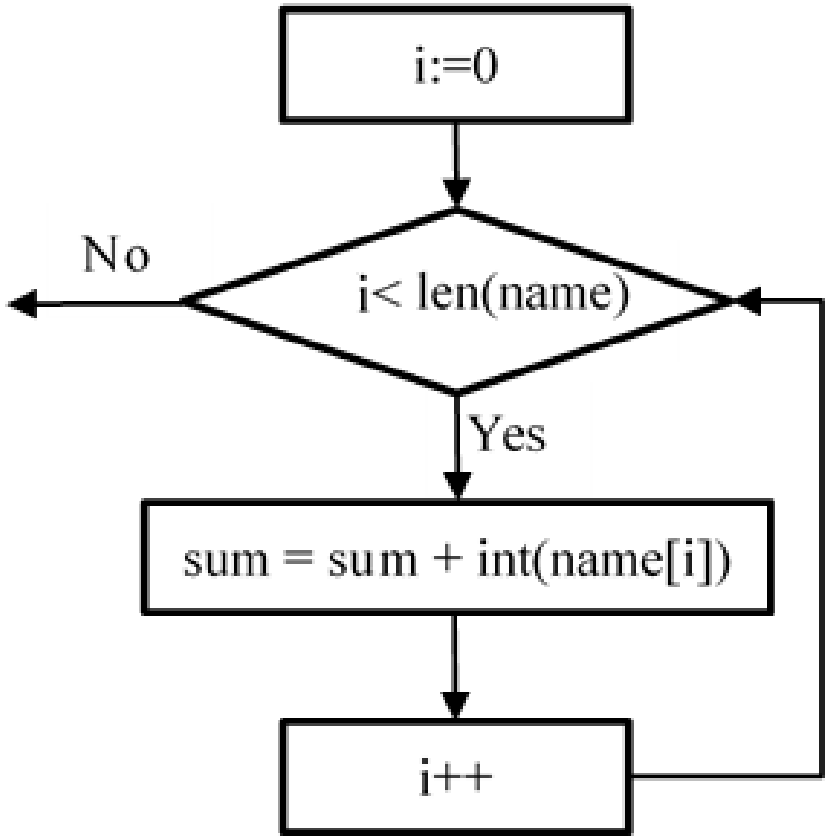
- 选择流
  - 根据条件，选择某一部分代码块执行
  - if……else……

```
if <simple statement>; <condition exp 1> {
    <code block 1>
} else if <condition exp 2> {
    <code block 2>
} ... else if <condition exp n-1> {
    <code block n-1>
} else {
    <code block n>
}
```

控制流

- 循环流
  - 根据条件，判断是否重复执行某个代码块





编程知识 —实验涉及的符号

符号	解释	例子
/、%	除法（除数不能为0）和取余。整数x和y之间的商q=x/y和余数r=x%y满足如下关系：x=q*y+r且|r|<|y|	5/3 值为1 5%3 值为2
++	递增，i++相当于i=i+1	Var i int = 1 i++ 值为2
==、!=	相等、不相等	1 == 1 值为true 1 != 1 值为false
=	赋值	var v int = 1 v = 3
:=	声明变量并赋值	i := 1 相当于 var i int = 1

简单操作

Println ：可以打印字符串和变量的值
Printf ： 只能打印字符串类型的变量，打印其他形式的变量需要在字符串中增加占位符

fmt.Printf(a)　//error

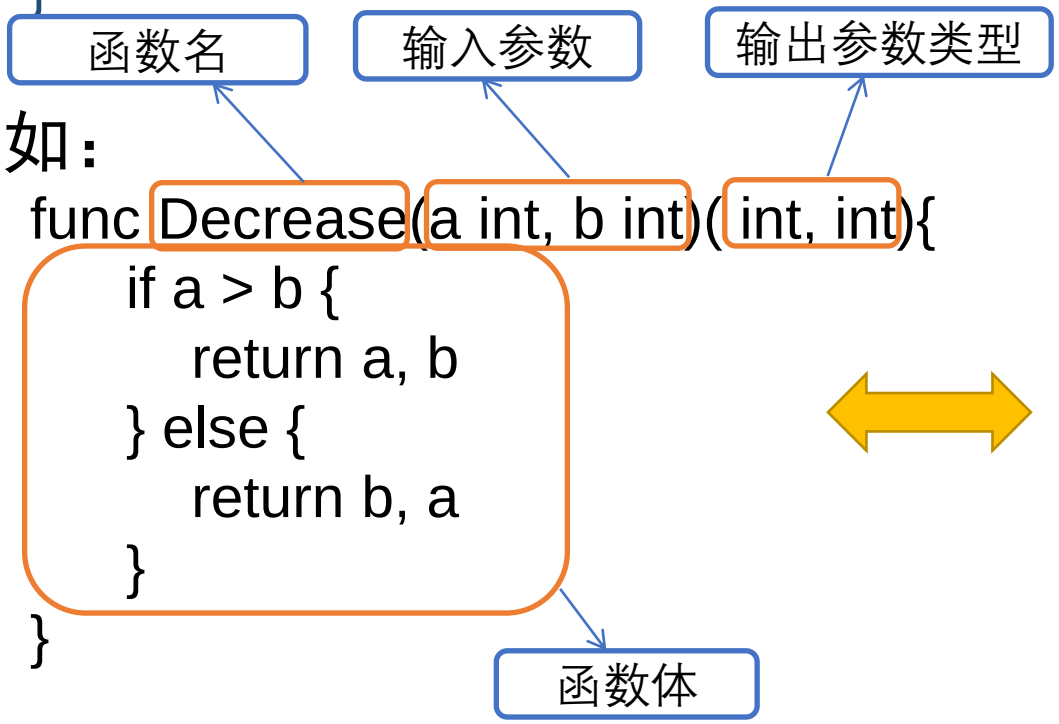
- 通过 fmt.Printf 函数的占位符机制，展示数值和数字符号的关系

```
package main
import "fmt"
func main() {
        fmt.Printf (“Decimal: %d\n”,63)        // 十进制表示
        fmt.Printf (“Hex: 0x%X\n”,63)          // 十六进制表示
        fmt.Printf (“Binary: %b\n”,63)         // 二进制表示
        fmt.Printf(“Character: %c\n”,63)       //  ASCII 码 63 对应的字符
        fmt.Printf(“String: %c%c\n”,63)                  // 出错
        fmt.Printf(“String: %c%c\n”,6,3)                 // 其他形式的代表6、3的乱码
        fmt.Printf(“String: %c%c\n”,6+‘0’,3+‘0’)         // 两个ASCII码字符 “6” “3”
}
```

接下来考虑：
怎样分别输出
6，3

函数的声明和调用

- 具体形式：
  - func <函数名>(<参数1名称，类型>, <参数2名称，类型>, …)(返回参数1类型，参数2类型，…){
    函数体
    }



- 例如：

```
func Decrease(a int, b int)( int, int){
    if a > b {
        return a, b
    } else {
        return b, a
    }
}
```



```
func Decrease(a, b int)( int, int) {
    if a > b {
        return a, b
    } else {
        return b, a
    }
}
```

函数调用

```
func main() {
    fmt.Println(Decrease(-1,3))
}
```