



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

计算机科学导论实验

编程基础-1

从hello到Name2Number.go
基本数据类型、基本语句、循环

zxu@ict.ac.cn
zhangjialin@ict.ac.cn

编程基础实验安排

周次	日期	实验课	课程内容	作业提交内容	截止时间
2	3月7日	编程基础-1	从hello到 Name2Number.go	name_to_number-0.go文件 与程序结果	2025/3/23 23:30
3	3月14日	编程基础-2	基本数据类型、基本语 句、循环	Name2Number.go文件与程序 结果	
			切片与递归 简版快速排序程序 fastsort.go	fastsort.go文件与程序中 间结果	
4	3月21日	编程基础-3	斐波那契计算机	RFC	

- 作业提交后会立刻返回成绩。
- 在截止时间前四次作业均可反复多次提交。

提纲

- 目标：知道编程入门知识
 - 编辑、编译、执行、调试的程序开发生命周期
 - 字面量、变量、内置函数调用、表达式
 - 程序结构、声明、赋值语句、字符串、数组、循环、打印语句
- 步骤
 - 在云计算环境重现hello.go
 - 理解并重现name_to_number-0.go，用你自己的姓名拼音
- 资料
 - Linux操作系统命令行界面常见命令（Shell Commands）
 - 英文文本字符集（ASCII Characters）
 - 占位符（Formatting Verbs）
 - 转义码（Escape Sequences）

云计算环境

- <https://course.things.ac.cn:10088/>

计算机科学导论 作业 实验 编程 ▾

1.

计算机科学导论 作业 实验 编程 ▾

2. 复制密码

密码:

3.

注意: 此密码仅用于登录平台, 不要泄

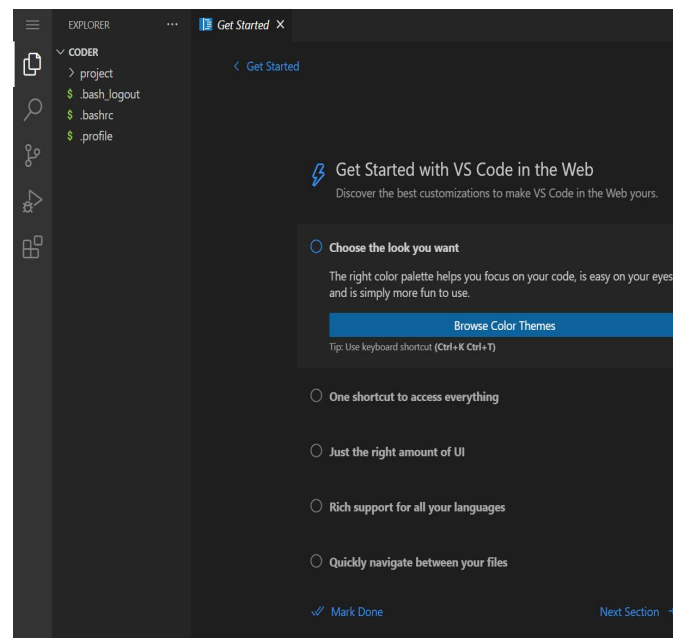
Welcome to code-server

Please log in below. Password was set from \$PASSWORD.

4. 粘贴密码

5.

6. 进入该界面



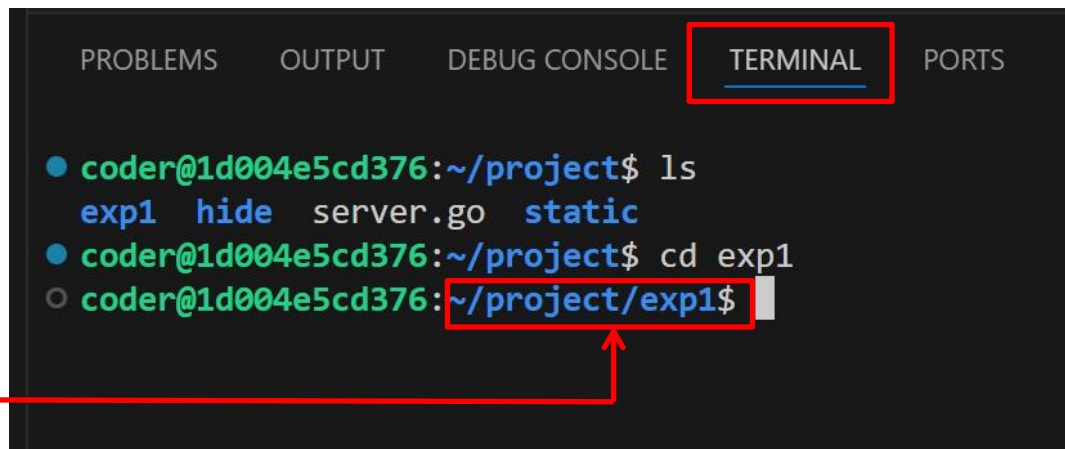
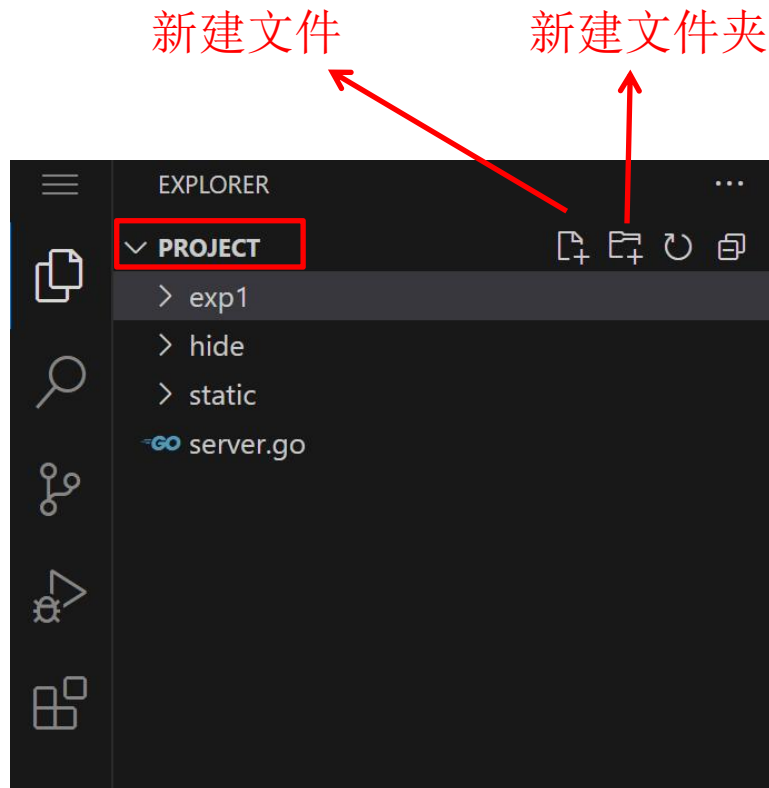
云计算环境

- 新建文件/文件夹

- 默认打开的是project文件夹
 - 请在project文件夹下创建新文件/文件夹
 - 否则不会被保存
- 点击新建文件夹(exp1)

- 打开终端 (TERMINAL)

- 快捷键 `ctrl + ``
- `ls` 列出目录内容
- `cd [dirname]` 切换目录
 - `cd exp1`
进入exp1目录
 - `cd ..` 返回上一级目录
 - 后续操作在 `project/exp` 目录下进行



1. 执行几个简单程序

● null.go

- 程序有两个语句（statement）
- 同学们写的程序属于主包（main package）
- 每个主包有一个主函数（main function）

```
package main      // main包
func main() {     // 定义了main函数
}                 // 它的函数体是空的
```

命令提示符
Command prompt

```
> code null.go
> go build null.go
> ./null
>
```

光标
cursor

编辑命令
编译命令
执行命令

Shell是命令
行解释器

1. 执行几个简单程序

● hello.go

- `fmt`是Go语言自带的**程序包**称为**库**（**library**）
 - 其他人开发了`fmt`供用户重用，`fmt` 包含函数`fmt.Println`
 - 注意：点号标记法（dot notation）
- 程序中的函数很像数学函数
 - 输入数据→输出数据
 - 但可包含**副作用**（**side effect**）
 - 此例`hello.go`程序的主函数只有副作用，即打印`hello!`

```
> go build hello.go
> ./hello
hello!
>
```

分别使用两条命令进行编译和执行

```
> go run hello.go
hello!
>
```

两个命令也可以合并成一个命令

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

```
// hello.go程序的主包
// 该程序导入一个库包fmt
// 该程序声明一个主函数
// 输出语句打印出hello!
```

本实验调用fmt软件包中的两个函数

```
package fmt          // The fmt package, 它不是主程序包, 而是库包
.....
func Println(...) ...{ // 定义了 Println 函数
...                  // 可以被其他程序以 fmt.Println 的方式调用
}                  // 点号标记法 (dot notation)
.....
func Printf(...) ...{ // 定义了 Printf 函数
...                  // 可以被其他程序以 fmt.Printf 的方式调用
}
.....
func Scanf(...) ...{  //定义了 Scanf 函数
...                  // 可以被其他程序以 fmt.Scanf 的方式调用
}
```

两个输出语句会输出相同的结果

```
fmt.Printf("hello! %d\n",63)
fmt.Println("hello!",63)
```

输出结果相同

```
> hello! 63
```

可能出现各种错误

- 将打印语句 `fmt.Println("hello!")` 替换成
 - `fmt.Printf("hello!\n")` 必须显式加换行符（即 `\n` 转义符）
 - `fmt.Printf("%c%c%c%c%c%c\n", XXX)`
 - 用对应 "hello!" 的正确的ASCII字符编码替换XXX

产生同样输出结果，即打印出**hello!**

- 重现编译错误

```
package main
import "fmt"
func main() {
    fmt.Printf("hello!\n")
    fmt.Printf("%c%c%c%c%c%c\n", 'h', "e", "l", "l", "o", '!')
}
```

可能出现各种错误

- 将打印语句 `fmt.Println("hello!")` 替换成
 - `fmt.Printf("hello!\n")` 必须显式加换行符（即 `\n` 转义符）
 - `fmt.Printf("%c%c%c%c%c%c\n", XXX)`
 - 用对应 "hello!" 的正确的ASCII字符编码替换XXX

产生同样输出结果，即打印出**hello!**

- 重现运行时错误

```
package main
import "fmt"
func main() {
    fmt.Printf("hello!\n")
    fmt.Printf("%c%c%c%c%c%c\n", "hello!")
}
```

可能出现各种错误

- 将打印语句 `fmt.Println("hello!")` 替换成
 - `fmt.Printf("hello!\n")` 必须显式加换行符（即 `\n` 转义符）
 - `fmt.Printf("%c%c%c%c%c%c\n", XXX)`
 - 用对应 "hello!" 的正确的ASCII字符编码替换XXX

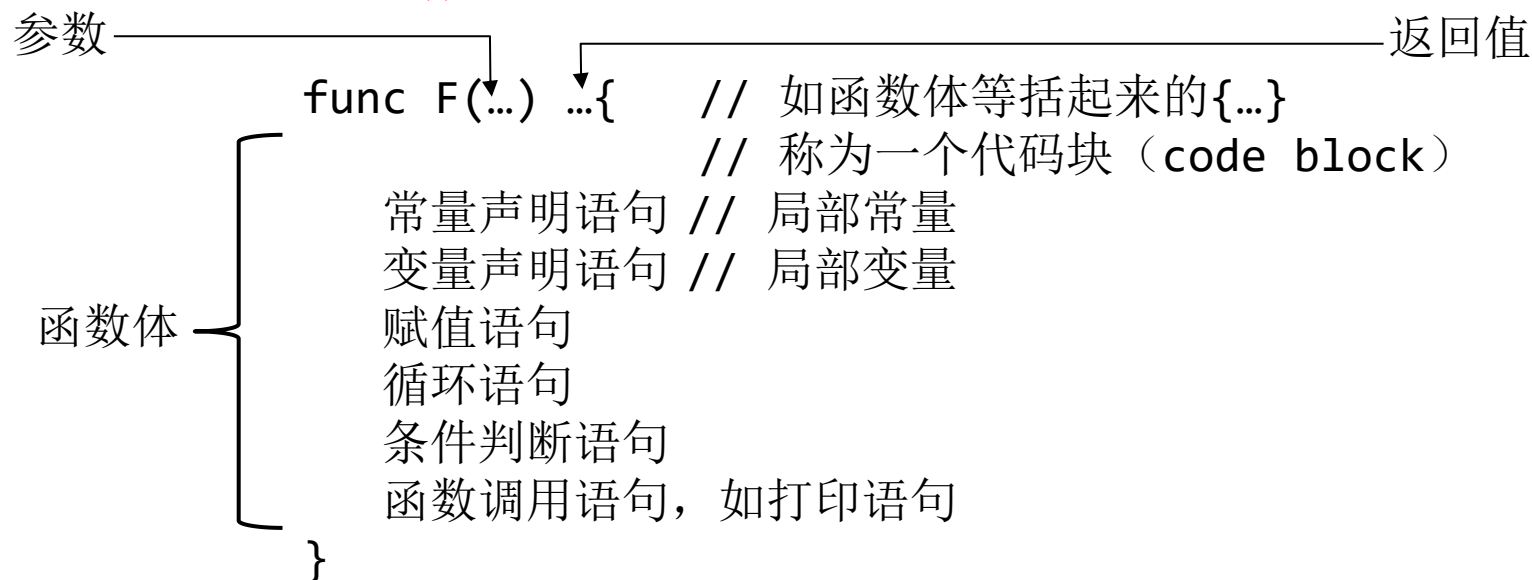
产生同样输出结果，即打印出**hello!**

- 正确代码为

```
package main
import "fmt"
func main() {
    fmt.Printf("hello!\n")
    fmt.Printf("%c%c%c%c%c%c\n", 104, 101, 108, 108, 111, 33)
}
```

Go程序的基本结构

主包声明语句	<code>package main</code>	// 定义主包（主程序包）
导入语句	<code>import ...</code>	// 导入其他人写的包，如不调用则不出现
常量声明语句	<code>const ...</code>	// 声明常量，可不出现
变量声明语句	<code>var ...</code>	// 声明变量，可不出现
函数声明语句	<code>func F(...) ...{...}</code>	// 声明程序员自定义的函数，可不出现
主函数声明语句	<code>func main() {</code>	// 声明主包中必须有的主函数
	<code>...</code>	// 主函数的函数体
	<code>}</code>	
	<code>main()</code>	// 缺省的主函数调用，不出现



求日出时间Golang程序中的 字面量、变量、内置函数调用、表达式

```
package main
import "fmt"
import "math"
func main() {
    length := 40000*math.Cos(40*3.14159/180) //指出函数调用、字面量、变量、表达式
    S := length / (24*3600)
    fmt.Println("自转速度S =",S, "千米/秒")    // 有多少个字面量？多少个变量？
    D = length * (116-76) / 360
    T = D / S
    fmt.Println("喀什日出时间 =", float64(B)+T, "秒")
    D = length * (116-93) / 360
    T = D / S
    fmt.Println("敦煌日出时间 =", float64(B)+T, "秒")
}
```

2.1 从字符串到数的变换

- 两个程序例子: "Alan Turing"变换到1045
 - 使用`%d`的简版程序 `name_to_number-0.go`
 - 使用`%c`的程序 `Name2Number.go`

- 新出现的变量类型与语句

- 变量类型

- 整数: `int`
 - 符号串: `string`
 - 字符数组

- 语句类型

- 赋值语句
 - 循环语句 (for loop)

```
package main          //name_to_number-0.go
import "fmt"
func main() {
    var name string = "Alan Turing"
    sum := 0
    for i := 0; i < len(name); i++ {
        sum = sum + int(name[i])
    }
    fmt.Printf("%d\n", sum)
}
```

"Alan Turing"的ASCII编码

= [65, 108, 97, 110, 32, 84, 117, 114, 105, 110, 103]

$D_6D_5D_4$ $D_3D_2D_1D_0$	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

name_to_number-0.go

- 符号串string是字符数组
- 声明变量的两种方式

```
var name string
name = "Alan Turing"
```

等价于

```
var name string = "Alan Turing"
```

基本等价于

```
name := "Alan Turing"
```

```
//name_to_number-0.go
package main
import "fmt"
func main() {
    var name string = "Alan Turing"
    sum := 0
    for i := 0; i < len(name); i++ {
        sum = sum + int(name[i])
    }
    fmt.Printf("%d\n", sum)
}
```

注意: name[4] 是空格' '=32

	A	l	a	n		T	u	r	i	n	g
name =	65	108	97	110	32	84	117	114	105	110	103
index →	0	1	2	3	4	5	6	7	8	9	10

len(name) is 11

name[0]='A'=65, name[1]='l'=108, name[2]='a'=97,
name[3]='n'=110, name[4]=' '=32, name[5]='T'=84,
name[6]='u'=117, name[7]='r'=114, name[8]='i'=105,
name[9]='n'=110, name[10]='g'=103.

如何将一个数组的元素相加？

- 问题: 将name[i]的11个元素加起来

- name = [65 108 97 110 32 84 117 114 105 110 103]

- 如何做到？

- 解决方案 1

```
sum := 0
sum = sum + name[0]
sum = sum + name[1]
sum = sum + name[2]
sum = sum + name[3]
sum = sum + name[4]
sum = sum + name[5]
sum = sum + name[6]
sum = sum + name[7]
sum = sum + name[8]
sum = sum + name[9]
sum = sum + name[10]
```



sum

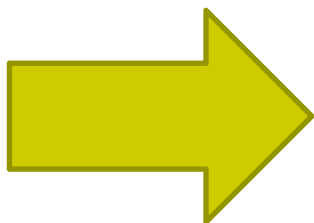
如何将一个数组的元素相加？

- 问题：将name[i]的11个元素加起来
- name = [65 108 97 110 32 84 117 114 105 110 103]

- 如何做到？

- 解决方案 1

```
sum := 0
sum = sum + name[0]
sum = sum + name[1]
sum = sum + name[2]
sum = sum + name[3]
sum = sum + name[4]
sum = sum + name[5]
sum = sum + name[6]
sum = sum + name[7]
sum = sum + name[8]
sum = sum + name[9]
sum = sum + name[10]
```



- 解决方案 2 为什么？

```
sum := 0
sum = sum + int(name[0])
sum = sum + int(name[1])
sum = sum + int(name[2])
sum = sum + int(name[3])
sum = sum + int(name[4])
sum = sum + int(name[5])
sum = sum + int(name[6])
sum = sum + int(name[7])
sum = sum + int(name[8])
sum = sum + int(name[9])
sum = sum + int(name[10])
```

name[0] = 65 = 01000001

```
sum := 0
sum = sum + int(name[0])
sum = sum + int(name[1])
...
sum = sum + int(name[10])
```

解决方案 2

name[0] = 65 = 01000001

类型转换操作

```
sum := 0
sum = sum + name[0]
sum = sum + name[1]
...
sum = sum + name[10]
```

Solution 1

```
sum := 0
sum = sum + int(name[0])
sum = sum + int(name[1])
...
sum = sum + int(name[10])
```

Solution 2

求和

[65 108 97 110 32 84 117 114 105 110 103]

name[1] = 108 = 01101100

[illegible]
$$\text{sum} := 0$$

```
sum = sum + int(name[0])
```

$$= 0 + 65 = 65$$
[illegible]

```
sum = sum + int(name[1])
```

$$= 65 + 108 = 173$$
[illegible]

解决方案2 的代码仍然有问题

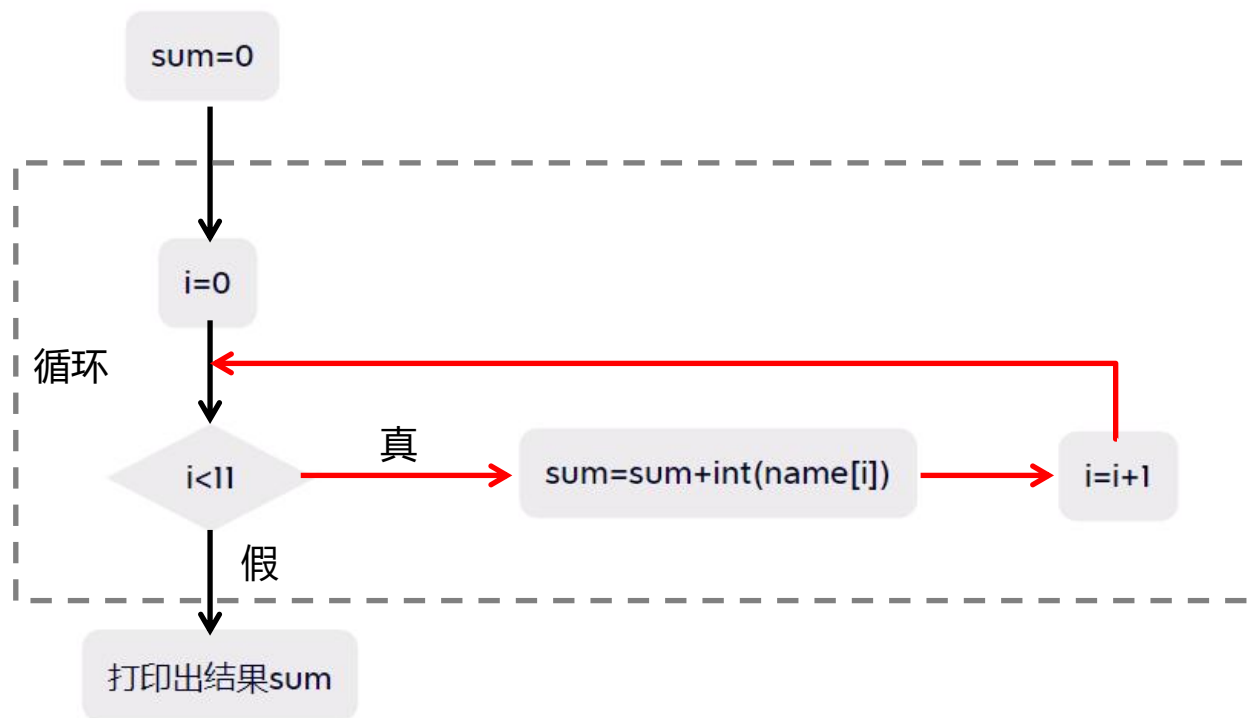
1. 与特定学生姓名绑定
 - 不适用于 `len(name) != 11` 的学生
2. 繁琐、重复的代码
3. 代码长度与问题规模成比例
 - 可能是不良设计的一个迹象
 - 如果 `len(name) == 1000000` 怎么办?
 - 违反了程序有限性原则
- Project 2: Turing Adder
 - 我们不希望图灵机的转移表大小与输入长度 N 成正比

```
sum := 0
sum = sum + int(name[0])
sum = sum + int(name[1])
sum = sum + int(name[2])
sum = sum + int(name[3])
sum = sum + int(name[4])
sum = sum + int(name[5])
sum = sum + int(name[6])
sum = sum + int(name[7])
sum = sum + int(name[8])
sum = sum + int(name[9])
sum = sum + int(name[10])
```

for loop循环抽象

- 初始化: 数组索引从零开始 $i = 0$
- 重复执行循环体, 每次迭代索引加1, 直到 $i \geq 11$

```
sum := 0
for i := 0; i < 11; i++ {
    sum = sum + int(name[i])
}
fmt.Printf("%d\n", sum)
```



幸好Go语言提供了for loop循环抽象

- 初始化: 数组索引从零开始 $i = 0$
- 重复执行循环体, 每次迭代索引加1, 直到 $i \geq 11$

```
package main
import "fmt"
func main() {
    var name string = "Alan Turing"
    sum := 0
    sum = sum + int(name[0])
    sum = sum + int(name[1])
    sum = sum + int(name[2])
    sum = sum + int(name[3])
    sum = sum + int(name[4])
    sum = sum + int(name[5])
    sum = sum + int(name[6])
    sum = sum + int(name[7])
    sum = sum + int(name[8])
    sum = sum + int(name[9])
    sum = sum + int(name[10])
    fmt.Printf("%d\n", sum)
}
```

```
package main
import "fmt"
func main() {
    var name string = "Alan Turing"
    sum = 0
    for i := 0; i < 11; i++ {
        sum = sum + int(name[i])
    }
    fmt.Printf("%d\n", sum)
}
```

循环抽象的诀窍: 综合**变**与**不变**
不变: for循环结构不变
变: 索引值变化, 累加值sum变化

理解并重现name_to_number-0.go

- 用你自己的姓名拼音，手算姓名编码
 - 如“Xu Zhi Wei”的姓名编码是861
 - $88+117+32+90+104+105+32+87+101+105 = 861$
- 用你自己的姓名拼音修改后程序，并向自己说明程序含义
 - 例如，当 $i=1$ 时，`sum = sum + int(name[i])`语句中下列值是什么？
 - 左边sum，右边sum，`int(name[i])`，`name[i]`
 - 为什么是`sum = sum + int(name[i])`，而不是`sum = sum + name[i]`
- 运行修改后程序，验证程序结果正确

```
package main      //name_to_number-0.go
import "fmt"
func main() {
    var name string = "Alan Turing" // 用你自己的姓名拼音
    sum := 0
    for i := 0; i < len(name); i++ {
        sum = sum + int(name[i])
    }
    fmt.Printf("%d\n", sum)
}
```

理解并重现name_to_number-0.go

- 程序中的关键字、作用域、全局变量、局部变量
 - 列出所有关键字
 - 列出变量的作用域: `name`, `sum`, `i`
 - 列出全局变量
 - 列出局部变量

```
package main      //name_to_number-0.go
import "fmt"
func main() {
    var name string = "Alan Turing" // 用你自己的姓名拼音
    sum := 0
    for i := 0; i < len(name); i++ {
        sum = sum + int(name[i])
    }
    fmt.Printf("%d\n", sum)
}
```

作业提交要求:

- 使用自己的姓名拼音重现的name_to_number-0.go文件
 - 注意: 替换成姓名的拼音请与国科大邮箱一致。例如邮箱是 zhangsan24@mails.ucas.ac.cn, 则姓名拼音是"Zhang San"
 - (每个字拼音首字母大写, 之间用1个空格隔开)
- 程序结果, 即你的姓名编码

作业提交链接:

https://course.things.ac.cn:10088/exp/basic_programming

作业提交截止时间:

[2025/3/23 23:30](#)

基础编程

第1次作业 ▾

请提交

• Name2Number.go 未选择文件

• 请输入你的姓名编码

- 作业提交后会立刻返回成绩。
- 在截止时间前四次作业均可反复多次提交。