Go入门

软件，goLang，学一门语言，好的编辑器，如虎添翼

》目录，就按一般的引入就可以

import "../core"

》要运行

一般 ，package main

》一次import多个

用小括号

入门知识点，备忘

cmd 进入项目的目录，不是bin的目录，

$ go run hello.go

注意：

func main() { // 错误，{ 不能在单独的行上，开头不可单独一行，结尾可以

声明变量

第三种，省略var, 注意 :=左侧的变量不应该是已经声明过的，否则会导致编译错误。

v\_name := value

// 例如var a int = 10var b = 10

c := 10

//这种不带声明格式的只能在函数体中出现//g, h := 123, "hello"

声明常量

const WIDTH int = 5

const a, b, c = 1, false, "str" //多重赋值

const (

a = "abc"

b = len(a)

c = unsafe.Sizeof(a)

)

循环

numbers := [6]int{1, 2, 3, 5}

for i,x:= range numbers {

fmt.Printf("第 %d 位 x 的值 = %d\n", i,x)

}

闭包

// 闭包使用方法

func add(x1, x2 int) func(x3 int,x4 int)(int,int,int) {

i := 0

return func(x3 int,x4 int) (int,int,int){

i++

return i,x1+x2,x3+x4

}

}

Go 语言变量作用域

作用域为已声明标识符所表示的常量、类型、变量、函数或包在源代码中的作用范围。

Go 语言中变量可以在三个地方声明：

函数内定义的变量称为局部变量

函数外定义的变量称为全局变量

函数定义中的变量称为形式参数

接下来让我们具体了解局部变量、全局变量和形式参数。

数组

var balance [10] float32

指针

## 什么是指针

一个指针变量指向了一个值的内存地址。

类似于变量和常量，在使用指针前你需要声明指针。指针声明格式如下：

var var\_name \*var-type

fmt.Printf("变量的地址: %x\n", &a )

var-type 为指针类型，var\_name 为指针变量名，\* 号用于指定变量是作为一个指针。以下是有效的指针声明：

var ip \*int /\* 指向整型\*/

var fp \*float32 /\* 指向浮点型 \*/

空指针判断：

if(ptr != nil) /\* ptr 不是空指针 \*/

if(ptr == nil) /\* ptr 是空指针 \*/

Go 语言结构体

结构体表示一项记录，比如保存图书馆的书籍记录，每本书有以下属性：

Title ：标题

Author ： 作者

Subject：学科

ID：书籍ID

结构体定义需要使用 type 和 struct 语句。

struct 语句定义一个新的数据类型，结构体有中有一个或多个成员。type 语句设定了结构体的名称。

package main

import "fmt"

type Books struct {

title string

author string

subject string

book\_id int}

func main() {

// 创建一个新的结构体

fmt.Println(Books{"Go 语言", "www.runoob.com", "Go 语言教程", 6495407})

// 也可以使用 key => value 格式

fmt.Println(Books{title: "Go 语言", author: "www.runoob.com", subject: "Go 语言教程", boo k\_id: 6495407}) // 忽略的字段为 0 或 空

fmt.Println(Books{title: "Go 语言", author: "www.runoob.com"})

}

输出结果为：

{Go 语言 www.runoob.com Go 语言教程 6495407}

{Go 语言 www.runoob.com Go 语言教 程 6495407}

{Go 语言 www.runoob.com 0}

结构体指针

定义指向结构体的指针类似于其他指针变量

var struct\_pointer \*Books

以上定义的指针变量可以存储结构体变量的地址。查看结构体变量地址，可以将 & 符号放置于结构体变量前：

struct\_pointer = &Book1;

使用结构体指针访问结构体成员，使用 "." 操作符：

struct\_pointer.title;

接下来让我们使用结构体指针重写以上实例：

package main

import "fmt"

type Books struct {

title string

author string

subject string

book\_id int}

func main() {

var Book1 Books /\* Declare Book1 of type Book \*/

var Book2 Books /\* Declare Book2 of type Book \*/

/\* book 1 描述 \*/

Book1.title = "Go 语言"

Book1.author = "www.runoob.com"

Book1.subject = "Go 语言教程"

Book1.book\_id = 6495407

/\* book 2 描述 \*/

Book2.title = "Python 教程"

Book2.author = "www.runoob.com"

Book2.subject = "Python 语言教程"

Book2.book\_id = 6495700

/\* 打印 Book1 信息 \*/

printBook(&Book1)

/\* 打印 Book2 信息 \*/

printBook(&Book2)}

func printBook( book \*Books ) {

fmt.Printf( "Book title : %s\n", book.title);

fmt.Printf( "Book author : %s\n", book.author);

fmt.Printf( "Book subject : %s\n", book.subject);

fmt.Printf( "Book book\_id : %d\n", book.book\_id);}

Go 语言切片(Slice)

Go 语言切片是对数组的抽象。

Go 数组的长度不可改变，在特定场景中这样的集合就不太适用，Go中提供了一种灵活，功能强悍的内置类型切片("动态数组"),与数组相比切片的长度是不固定的，可以追加元素，在追加时可能使切片的容量增大。

切片不需要说明长度。

或使用make()函数来创建切片

切片初始化

s :=[] int {1,2,3 }

直接初始化切片，[]表示是切片类型，{1,2,3}初始化值依次是1,2,3.其cap=len=3

s := arr[:]

len() 和 cap() 函数

len() 方法获取长度

 cap() 可以测量切片最长可以达到多少

空(nil)切片

if(numbers == nil){

fmt.Printf("切片是空的")

}

切片截取

numbers[:3]

append() 和 copy() 函数

Go 语言范围(Range)

Go 语言中 range 关键字用于 for 循环中迭代数组(array)、切片(slice)、通道(channel)或集合(map)的元素。