go区块链课程 讲师:张长志

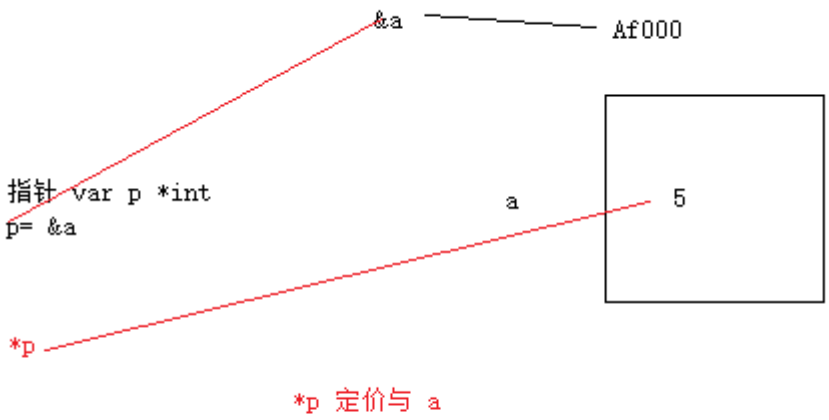
类型	名称	长度	默认值	说明
pointer	指针（地址）		nil	
array	数组		0	
slice	切片		nil	引用类型
map	字典		nil	引用类型
struct	结构体			

go 语言里面对指针做了弱化

指针

保存某个变量的地址，需要指针类型 *int 保存int类型的地址

声明（定义），定义只是特殊的声明，定义一个变量p 类型为*int var p *int



*p 定价与 a

scanner（&p）

代码实战

```
package main

import "fmt"

func main(){

    var a int = 10
    //每个变量有2层含义：变量的内存 变量地址
    fmt.Printf("a=%d\n",a) //变量的内存
    fmt.Printf("&a = %v\n",&a) //变量的地址

    //保存地址必须要用指针（地址）

    var p *int  //定义一个变量p，类型为int类型
    p = &a  //指针变量指向谁，就把谁的地址赋值给指针变量
    fmt.Printf("p=%v,&a=%v,*p=%v\n",p,&a,*p)

    *p = 1000
    fmt.Printf("p=%v,&a=%v,*p=%v\n",p,&a,*p)
}
```

指针的注意点

不能操作没有合法指向的内存

```
import "fmt"

func main(){
    var p *int //指针变量
    p = nil
    fmt.Println("p=",p)

    //*p = 1000 invalid memory address or nil pointer dereference
    //因为p没有任何指向

    var a int
    p = &a
    *p = 1000
    fmt.Println("a=",a)

}
```

new 开辟空间

```
package main

import "fmt"

func main(){
     //a := 10     //整形变量
     var p *int //指针变量
    // p = &a
    // *p = 1000 //a = 1000

    p   = new(int)
    fmt.Printf("p=%v,*p=%d\n",p,*p)

    *p = 6000

    fmt.Printf("p=%v,*p=%d\n",p,*p)

    q := new(int) //自动推导
    *q =88888
    fmt.Println("*q=",*q)
}
```

普通变量和指针变量作为参数传递

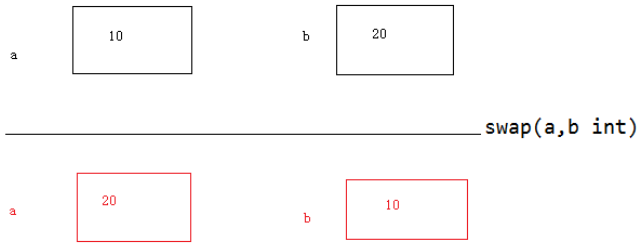
1、普通参数

```
func swap(a,b int)  {
    a,b = b,a
    fmt.Printf("swap:a=%d,b=%d\n",a,b) //20 10
}

func main(){
    a,b := 10,20
    //通过一个函数swap 交换a b的内容
    swap(a,b)
    fmt.Printf("main: a=%d,b=%d\n",a,b) //10 20
}
```

```go
func swap(a,b int)  {
    a,b = b,a
    fmt.Printf("swap:a=%d,b=%d\n",a,b) //20
10
}

func main(){
    a,b := 10,20
    //通过一个函数swap 交换a b的内容
    swap(a,b)
    fmt.Printf("main: a=%d,b=%d\n",a,b) //10
20
}
```



2.指针作为函数

```go
import "fmt"

func swap1(p1,p2 *int){
   *p1,*p2 = *p2,*p1
}

func main()  {
   a,b :=10,20
   swap1(&a,&b)

   fmt.Printf("main: a= %d,b=%d\n",a,b)

}
```

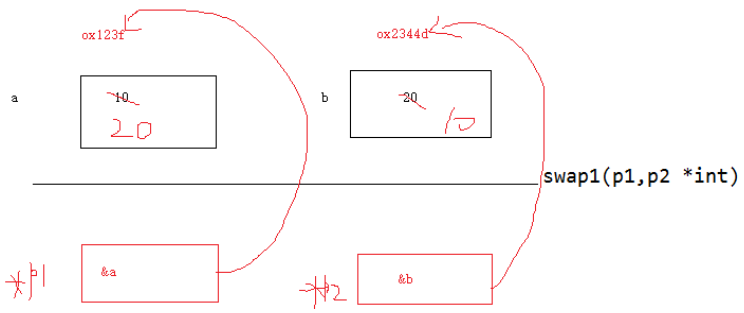
```go
import "fmt"

func swap1(p1,p2 *int){
   *p1,*p2 = *p2,*p1
}

func main()  {
    a,b :=10,20
    swap1(&a,&b)

    fmt.Printf("main: a= %d,b=%d\n",a,b)

}
```



数组

i1 =100 i2 =95

数组就是指**一系列同一类型**的***数据集合***。数组中包含的每个数据被称为数组元素(element),一个数组包含的元素个数被称为数组长度

var id [50]int

id[0] = 100

id[1] = 100

id[2] =99

```
package main

import "fmt"

func main(){
   //id1 := 1
   //id2 :=2
   //id3 :=3
   var id [50]int  //数组，同一类型的集合

   for i :=0 ;i<len(id);i++{
      id[i] = i+1
      fmt.Printf("id[%d]=%d\n",i,id[i])
   }
}
```

1数组基本使用

```
package main

import (
   "fmt"
)

func main(){
   //定义一个数组 [10]int和 [5]int 是不同的类型
   //[数字]，这个数字代表的是数组的元素个个数
   var a [10]int
   var b [5]int
   fmt.Printf("len(a)=%d,len(b)=%d\n",len(a),len(b))

   //注意点 定义数组，指定的数组的个数必须是常量 non-constant array bound n

   //n:=10
   //var c [n]int
   //fmt.Printf("len(c)=%d\n",len(c))
   // 操作数组元素，从0开始，到len()-1,不对称元素，这个数字叫下标,下标是可以使用变量的
   a[0]= 1
   i :=1
   a[i] = 10

   //赋值，每个元素
   for i :=0;i<len(a);i++{
```

```
        a[i]=i+1
    }

   //打印，第一个返回下标 ，第二个是返回元素对应的值
    for i,data :=range a{
        fmt.Printf("a[%d]=%d\n",i,data)
    }

}
```

2.数组的初始化

```
package main

import (
    "fmt"
)

func main()  {
     //声明定义同时初始化

    var a [5]int = [5]int{1,2,3,4,5}
    fmt.Println("a=",a)

    b := [5]int{1,2,3,4,5} //自动推导
    fmt.Println("b=",b)

    //部分初始化 没有初始化的int自动补全0
    c :=[5]int{1,2,3}
    fmt.Println("c=",c)

    //指定某个元素的初始化
    d := [5]int{2:10,4:20}
    fmt.Println("d=",d)

}
```

二维数组

go语言班级 50 c语言班级 40 大数据班级 30

[][] 有多少个【】 代表有多少维

有多少个【】就用多个循环

```
var a[3][4]int
```

```go
package main

import "fmt"

func main(){
   //有多少个【】 代表有多少维
   //有多少个【】就用多个循环

   var a [3][4]int //有三个班级 每个班级有4人 1 ，2,3，4 5，6，7,8 9,10,11,12
   k:=0
   for i:=0;i<3;i++{
      for j:=0;j<4;j++{
         k++
         a[i][j] = k
         fmt.Printf("a[%d][%d]=%d,",i,j,a[i][j])
      }
      fmt.Println()
   }

   fmt.Println("a=",a)

   //有3个元素 每个元素又是一维数组[4]int
   b := [3][4]int{{1,2,3,4},{5,6,7,8},{9,10,11,12}}
   fmt.Println("b=",b)

   //部分初始化，没有初始化的如果是int 补全0
   c := [3][4]int{{1,2,3},{2,6},{4}}
   fmt.Println("c=",c)

   d :=[3][4]int{{1,2,3,4}}
   fmt.Println("d=",d)

   e := [3][4]int{1:{5,6,7,8}}
   fmt.Println("e=",e)

}
```

数组的比较和赋值

```go
package main

import "fmt"

func main()  {
   //只支持 == ！= 比较是不是每个元素都一样 2个数组比较类型要一致 [3]int [4]int
   a := [5]int{1,2,3,4,5}
   b := [5]int{1,2,3,4,5}
   c:=[5]int{1,2,3}

   fmt.Println("a==b",a==b)
```

```
    fmt.Println("a==c",a==c)
   //同类型的数组可以赋值
    var d [5]int
    d = c
    fmt.Println("d=",d)


}
```

随机数使用

验证码 4 6 数 随机的

```
rand= 5951029679893600903
rand= 4258449715249457740
rand= 5697541505805793 87
rand= 5655783816535717285
rand= 2303413980504733195
rand= 79966101389923097 89
```

```
rand= 5951029679893600903
rand= 4258449715249457740
rand= 569754150580579387
rand= 5655783816535717285
rand= 2303413980504733195
rand= 7996610138992309789
```

```
package main

import (
    "math/rand"
    "fmt"
    "time"
)

func main()  {
    //如果种子参数一致，每次运行的程序产生的结果是一致的
    //每次运行让种子参数不一致
    //rand.Seed(6666)
    rand.Seed(time.Now().UnixNano())// 以当前系统时间作为种子参数

    for i:=0;i<6;i++{
        //fmt.Println("rand=",rand.Int()) //随机数很大
        fmt.Println("rand=",rand.Intn(100)) //限制在100以内的数字
    }
```

```
}
```

更多免费资料和视频：http://www.dataxueyuan.com/