数组Array

- 定义数组的格式：var <varName> [n]<type> ，n>=0
- 数组长度也是类型的一部分，因此具有不同长度的数组为不同类型
- 注意区分指向数组的指针和指针数组
- 数组在Go中为值类型
- 数组之间可以使用==或!=进行比较，但不可以使用<或>
- 可以使用new来创建数组，此方法返回一个指向数组的指针
- Go支持多维数组

Go语言版冒泡排序

Array 语法例子：

```go
package main

import (
    "fmt"
)

func main() {
    var a [2]int
    var b [1]int
    b = a
    fmt.Println(b)
}
```

```
/usr/local/go/bin/go build -i [/home/jiemin/code/GOlang/go/src/array]
# _/home/jiemin/code/GOlang/go/src/array
./array.go:10: cannot use a (type [2]int) as type [1]int in assignment
错误: 进程退出代码 2.
```

因为在go语言当中数组并不是一个统一的类型，而它将这个数组的长度也做为整个类型的一部分，所以说长度为2的int型数组和长度为1的int型数组，它是俩个不同长度的类型，也就是说，a和b之间是不能够直接等号去赋值，必须使用循环来完成赋值的操作，所以说这样是非法的。

```go
package main

import (
    "fmt"
)

func main() {
```

```
    var a [2]int
    var b [2]int
    b = a
    fmt.Println(b)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
[0 0]
成功: 进程退出代码 0.
```

这样的，就是合法的

改为string类型

```
package main

import (
    "fmt"
)

func main() {
    var a [2]string
    var b [2]string
    b = a
    fmt.Println(b)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
[ ]
成功: 进程退出代码 0.
```

```
package main

import (
    "fmt"
)

func main() {
    var a [2]string
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
[ ]
成功: 进程退出代码 0.
```

简写：

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[0 0]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1, 2}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[1 2]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[1 0]
成功: 进程退出代码 0.
```

大括号里面是1，说明长度为2的数组，字面值不够，就用零值补充。

使用长度为20的数组，前面19个数值让为零值，第二十个数值为1，那么就是用索引的方式

```go
package main

import (
    "fmt"
)

func main() {
    a := [20]int{19: 1}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
成功: 进程退出代码 0.
```

数组的长度(元素) 可以使用三个点...，让go语言自己去计算数组的长度

```go
package main

import (
    "fmt"
)

func main() {
    a := [...]int{1, 2, 3, 4, 5}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[1 2 3 4 5]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [...]int{0: 1, 1: 2, 2: 3}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
[1 2 3]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [...]int{19: 1}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
成功: 进程退出代码 0.
```

要区分指向数组的指针 和 指针数组
指向数组的指针：

```go
package main

import (
    "fmt"
)

func main() {
    a := [...]int{19: 1}
    var p *[20]int = &a
    fmt.Println(p)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array [/home/jiemin/code/GOlang/go/src/array]
&[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
成功: 进程退出代码 0.
```

前面多了一个取地址的符号 & ，表明这样的 p 它是取这样数组的一个地址。

指针数组:

```go
package main

import (
    "fmt"
)
```

```
6.
7.  func main() {
8.      x, y := 1, 2
9.      a := [...]*int{&x, &y}
10.     fmt.Println(a)
11. }
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[0xc42000a340 0xc42000a348]
成功: 进程退出代码 0.
```

a 保存的元素是指向int型的指针，然后取x，y的值，之后在输出a，就会输出变量x，变量y的地址，也就是说它实际上是保存了俩个变量的指针，并没有保存实际的值。

```
1.  package main
2.
3.  import (
4.      "fmt"
5.  )
6.
7.  func main() {
8.      a := [2]int{1, 2}
9.      b := [2]int{1, 2}
10.     fmt.Println(a == b)
11. }
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
true
成功: 进程退出代码 0.
```

```
1.  package main
2.
3.  import (
4.      "fmt"
5.  )
6.
7.  func main() {
8.      a := [2]int{1, 2}
9.      b := [2]int{1, 2}
10.     fmt.Println(a != b)
11. }
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
false
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1, 2}
    b := [2]int{1, 2}
    fmt.Println(a > b)
}
```

```
/usr/local/go/bin/go build -i [/home/jiemin/code/GOlang/go/src/array]
# _/home/jiemin/code/GOlang/go/src/array
./array.go:10: invalid operation: a > b (operator > not defined on array)
错误: 进程退出代码 2.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1, 2}
    b := [2]int{1, 2}
    fmt.Println(a < b)
}
```

```
/usr/local/go/bin/go build -i [/home/jiemin/code/GOlang/go/src/array]
# _/home/jiemin/code/GOlang/go/src/array
./array.go:10: invalid operation: a < b (operator < not defined on array)
错误: 进程退出代码 2.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1, 2}
    b := [1]int{1, 2}
    fmt.Println(a == b)
}
```

```
/usr/local/go/bin/go build -i [/home/jiemin/code/GOlang/go/src/array]
# _/home/jiemin/code/GOlang/go/src/array
./array.go:9: array index 1 out of bounds [0:1]
错误: 进程退出代码 2.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2]int{1, 2}
    b := [1]int{1}
    fmt.Println(a == b)
}
```

```
/usr/local/go/bin/go build -i [/home/jiemin/code/GOlang/go/src/array]
# _/home/jiemin/code/GOlang/go/src/array
./array.go:10: invalid operation: a == b (mismatched types [2]int and [1]int)
错误: 进程退出代码 2.
```

可以使用new关键字来获取数组的地址，等于使用 & 符号
使用new这个关键字呢，就会返回一个指向数组的指针，它可以直接使用索引对这个数字进行操作

```go
package main

import (
    "fmt"
)

func main() {
    a := new([10]int)
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
&[0 0 0 0 0 0 0 0 0 0]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
```

```
)

func main() {
    a := [10]int{}
    a[1] = 2
    fmt.Println(a)
    p := new([10]int)
    p[2] = 2
    fmt.Println(p)
}
```

```
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[0 2 0 0 0 0 0 0 0 0]
&[0 0 2 0 0 0 0 0 0 0]
成功: 进程退出代码 0.
```

多维数组

```
package main

import (
    "fmt"
)

func main() {
    a := [2][3]int{}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[[0 0 0] [0 0 0]]
成功: 进程退出代码 0.
```

a := [2][3]int{} 的意思就是数组的元素有俩个，每个元素又是长度为3的int型数组

```
package main

import (
    "fmt"
)

func main() {
    a := [2][3]int{{1, 1, 1}, {2, 2, 2}}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[[1 1 1] [2 2 2]]
成功: 进程退出代码 0.
```

```go
package main

import (
    "fmt"
)

func main() {
    a := [2][3]int{{1: 1}, {2: 2}}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[[0 1 0] [0 0 2]]
成功: 进程退出代码 0.
```

不知道顶级的元素有多少个，也可以使用三个点...来计算。三个点只能使用在最顶级，非顶级的不能三个点让它自己计算

```go
package main

import (
    "fmt"
)

func main() {
    a := [...][3]int{{1: 1}, {2: 2}}
    fmt.Println(a)
}
```

```
成功: 进程退出代码 0.
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[[0 1 0] [0 0 2]]
成功: 进程退出代码 0.
```

演示go语言版本的帽泡排序：

```go
package main

import (
    "fmt"
)

```

```go
func main() {
    a := [...]int{1, 2, 3, 4, 5, 6, 7, 8, 9}
    fmt.Println(a)

    l := len(a)
    for i := 0; i < l; i++ {
        for j := i + 1; j < l; j++ {
            if a[i] < a[j] {
                temp := a[i]
                a[i] = a[j]
                a[j] = temp
            }
        }
    }
    fmt.Println(a)
}
```

```
/home/jiemin/code/GOlang/go/src/array/array  [/home/jiemin/code/GOlang/go/src/array]
[1 2 3 4 5 6 7 8 9]
[9 8 7 6 5 4 3 2 1]
成功: 进程退出代码 0.
```