```
package main

import (
	"fmt"
	"sync"
)

var (
	oddWait sync.WaitGroup//oddWait 信号量，控制奇数消费者线程运行
	evenWait sync.WaitGroup//evenWait 信号量，控制偶数消费者线程运行
	wg sync.WaitGroup//控制 main 函数等待全部 goroutines 执行完后结束
)

/*
生产者线程
输入：奇数 channel，偶数 channel
处理：向奇数 channel 输入(1,3,5,7,9)，向偶数 channel 输入(0,2,4,6,8,)
 */
func producer(odd chan<- int, even chan<- int) {
	defer wg.Done()
	//wg 信号量 v(1)
	for i := 0; i < 10; i++ {
		if i%2 == 0 {
			even <- i
		} else {
			odd <- i
		}
	}
	close(odd)
	close(even)
	//关闭 channel
}

func oddConsunmer(odd <-chan int) {
	defer wg.Done()
	//wg 信号量 v(1)
	for {
		value, ok := <-odd
		if !ok {
			fmt.Printf("Consumer odd,work done\n")
			return
		}
		oddWait.Wait()//oddWait 信号量 p(1)
		fmt.Printf("Consumer odd, get value %d\n", value)
```

```
            oddWait.Add(1)//oddWait 信号量设值 1
            evenWait.Done()//evenWait 信号量 v(1)
        }
}

func evenConsunmer(even <-chan int){
        defer wg.Done()
        //wg 信号量 v(1)
        for {
            value, ok := <-even
            if !ok {
                fmt.Printf("Consumer even,work done\n")
                return
            }
            evenWait.Wait()//evenWait 信号量 p(1)
            fmt.Printf("Consumer even, get value %d\n", value)
            evenWait.Add(1)//evenWait 信号量设值 1
            oddWait.Done()//oddWait 信号量 v(1)
        }
}

func main(){
        oddChannel := make(chan int)
        evenChannel := make(chan int)
        //创建奇偶数 channel

        wg.Add(3)//wg 等待组设值为 3
        oddWait.Add(1)//oddWait 信号量初始值为 1

        go producer(oddChannel, evenChannel)
        go oddConsunmer(oddChannel)
        go evenConsunmer(evenChannel)
        //执行 go 线程

        wg.Wait()
        //wg 信号量 p(3)

}
```

执行结果

```
Consumer even, get value 0
Consumer odd, get value 1
Consumer even, get value 2
Consumer odd, get value 3
Consumer even, get value 4
Consumer odd, get value 5
Consumer even, get value 6
Consumer odd, get value 7
Consumer even, get value 8
Consumer even,work done
Consumer odd, get value 9
Consumer odd,work done

Process finished with exit code 0
```

程序思路

控制 evenConsunmer 和 oddConsunmer 交替执行输出语句可通过信号量实现。程序中分别声明了控制 evenConsunmer 执行的 evenWait 和控制 oddConsunmer 执行的 oddWait。通过执行 sync.WaitGroup.wait()（P 操作）和 sync.WaitGroup.done()（V 操作）便可实现线程间同步。

由于 evenConsunmer 最先执行，因此 evenWait 初值设为 0，当 evenConsunmer 执行 wait 操作时，由于没有需要等待的资源，因此可直接执行。oddConsunmer 后执行，因此设置 oddWait.Add(1)，这样即使 oddConsunmer 先到达，也会因为等待资源释放而阻塞。

之后当 evenConsunmer 执行完一条输出语句后，需要分别执行 evenWait.Add(1)和 oddWait.Done()，注意这两条语句顺序不能颠倒，如果 oddWait.Done()先执行，则 oddConsunmer 线程则有可能先执行 evenWait.done()，而 evenConsunmer 后执行 evenWait.Add(1)从而导致死锁。