



**«Московский государственный технический университет
имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

О т ч е т

По лабораторной работе № 5

**Название лабораторной работы: Основы асинхронного
программирования на Golang**

Дисциплина: Языки интернет программирования

Студент гр. ИУ6-32Б _____ Суворов Вакао А.
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Задание work

Код:

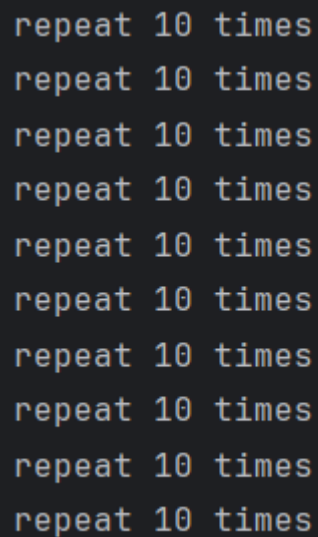
```
package main

import (
    "fmt"
    "sync"
    "time"
)

func work() {
    time.Sleep(1 * time.Second)
    fmt.Println("repeat 10 times")
}

func main() {
    wg := new(sync.WaitGroup)
    for i := 0; i < 10; i++ {
        wg.Add(1)
        go func(wg *sync.WaitGroup) {
            defer wg.Done()
            work()
        }(wg)
    }
    wg.Wait()
}
```

Результат (рис. 1)



```
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
repeat 10 times
```

Рис. 1

Задание pipeline

Код:

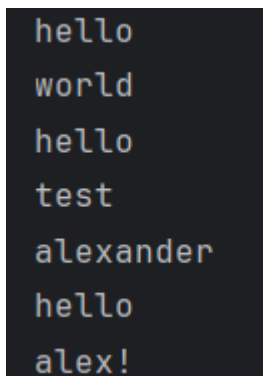
```
package main

import "fmt"

func removeDuplicates(in, out chan string) {
    var a string
    for i := range in {
        if i != a {
            out <- i
            a = i
        }
    }
    close(out)
}

func main() {
    inputChan := make(chan string)
    outputChan := make(chan string)
    go func() {
        inputChan <- "hello"
        inputChan <- "world"
        inputChan <- "hello"
        inputChan <- "hello"
        inputChan <- "test"
        inputChan <- "alexander"
        inputChan <- "hello"
        inputChan <- "alex!"
        close(inputChan)
    }()
    go removeDuplicates(inputChan, outputChan)
    for s := range outputChan {
        fmt.Println(s)
    }
}
```

Тестируем (рис. 2)



```
hello
world
hello
test
alexander
hello
alex!
```

Рис. 2

Задание calculator

Код:

```
package main

import "fmt"
```

```

func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan
struct{}) <-chan int {
    ans := make(chan int)
    go func(a chan int) {
        defer close(a)
        select {
            case b := <-firstChan:
                a <- b * b
            case d := <-secondChan:
                a <- d * 3
            case <-stopChan:
                return
        }
    }(ans)
    return ans
}

func main() {
    first, second := make(chan int), make(chan int)
    stopChan := make(chan struct{})
    calc := calculator(first, second, stopChan)
    first <- 4
    //second <- 2
    close(stopChan)
    fmt.Println(<-calc)
}

```

Результат (рис. 3)

16

Рис. 3

Вывод: разобралась с асинхронным программированием в Go