



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«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)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01

## О Т Ч Е Т

по лабораторной работе № 5

Название: Основы асинхронного программирования на Golang

Дисциплина: Языки интернет программирования

Студент

ИУ6-33Б

(Группа)

Б.Ш.

Истамилов

(И.О. Фамилия)

(Подпись, дата)

Преподаватель

В.Д. Шульман

(И.О. Фамилия)

(Подпись, дата)

Москва, 2024

Цель работы: изучение основ асинхронного программирования с использованием языка Golang.

Задание 1: Напишите элемент конвейера (функцию), что запоминает предыдущее значение и отправляет значения на следующий этап конвейера только если оно отличается от того, что пришло ранее.

Код программы:

```
package main

import (
    "fmt"
)

func removeDuplicates(inputStream chan string, outputStream chan string) {
    /*
    if temp != <-inputStream {
        temp = <-inputStream
        outputStream <- temp
    }
    */
    var temp string
    for v := range inputStream {
        if temp != v {
            temp = v
            outputStream <- temp
        }
    }
    close(outputStream)
}

func main() {

    channel := make(chan string, 10)
    channel1 := make(chan string, 10)

    for _, r := range "1222345567" {
        channel <- string(r)
    }

    close(channel)

    go removeDuplicates(channel, channel1)

    for v := range channel1 {
        fmt.Println(v)
    }

}
```

Результат работы программы:

```
balu@balu-B550M-DS3H:~/Рабочий стол/lr5$ go run main.go
1
2
3
4
5
6
7
```

Рисунок 1 — результат работы программы

Задание 2: Внутри функции main (функцию объявлять не нужно), вам необходимо в отдельных горутинах вызвать функцию work() 10 раз и дождаться результатов выполнения вызванных функций.

Код программы:

```
package main
```

```
import (  
    "fmt"  
    "sync"  
)
```

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

```
func work() {  
    fmt.Println("work is done")  
}
```

Результат работы программы:

```
● balu@balu-B550M-DS3H:~/Рабочий стол/lr5q2$ go run main.go
work is done
work is done
work is done
work is done
work is done
work is done
work is done
work is done
work is done
work is done
work is done
work is done
```

Рисунок 2 — результат работы программы

Задание 3: Функция получает в качестве аргументов 3 канала, и возвращает канал типа `<-chan int`.

- в случае, если аргумент будет получен из канала `firstChan`, в выходной (возвращенный) канал вы должны отправить квадрат аргумента.
- в случае, если аргумент будет получен из канала `secondChan`, в выходной (возвращенный) канал вы должны отправить результат умножения аргумента на 3.
- в случае, если аргумент будет получен из канала `stopChan`, нужно просто завершить работу функции.

Функция `calculator` должна быть неблокирующей, сразу возвращая управление. Ваша функция получит всего одно значение в один из каналов - получили значение, обработали его, завершили работу.

После завершения работы необходимо освободить ресурсы, закрыв выходной канал, если вы этого не сделаете, то превысите предельное время работы.

Код программы:

```
package main

import (
    "fmt"
)

type fm struct{}

func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
    ch4 := make(chan int)

    go func(chh chan int) {
        defer close(chh)
        select {
```

```

case x := <-firstChan:
ch4 <- x * x

case x := <-secondChan:
ch4 <- x * 3

case <-stopChan:
}

}(ch4)

return ch4
}

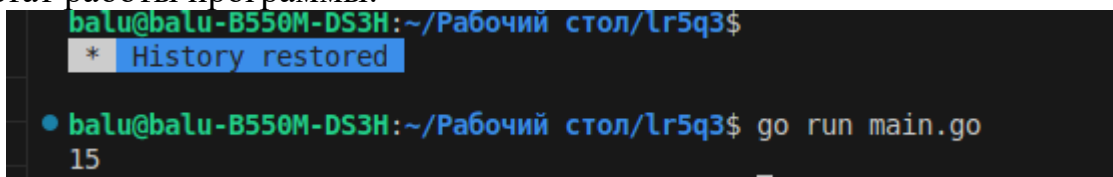
func main() {

ch1 := make(chan int)
ch2 := make(chan int)
ch3 := make(chan struct{})
go func() {
ch2 <- 5
}()
fmt.Println(<-calculator(ch1, ch2, ch3))

}

```

Результат работы программы:



```

balu@balu-B550M-DS3H:~/Рабочий стол/lr5q3$
* History restored

balu@balu-B550M-DS3H:~/Рабочий стол/lr5q3$ go run main.go
15

```

Рисунок 3 — результат работы программы

Вывод: в процессе выполнения лабораторной работы были освоены основы асинхронного программирования на языке go lang

Источники:

<https://stepik.org/course/54403/syllabus>