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 Компьютерные системы и сети**

О Т Ч Е Т

по лабораторной работе № 5

Название: Основы асинхронного программирования на Golang

Дисциплина: **Языки Интернет-Программирования**

Студент ИУ6-31Б _____ К.Д. Коротаев
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2024

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

Задание 1: Внутри функции main (функцию объявлять не нужно), вам необходимо в отдельных горутинах вызвать функцию work() 10 раз и дождаться результатов выполнения вызванных функций. Функция work() ничего не принимает и не возвращает.

Задание 2: Напишите элемент конвейера (функцию), что запоминает предыдущее значение и отправляет значения на следующий этап конвейера только если оно отличается от того, что пришло ранее.
Ваша функция должна принимать два канала - inputStream и outputStream, в первый вы будете получать строки, во второй вы должны отправлять значения без повторов. В итоге в outputStream должны остаться значения, которые не повторяются подряд.

Задание 3: Вам необходимо написать функцию calculator следующего вида:
func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int
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.

Ход работы:
Задание 1
Код main.go

main.go

C: > Users > admin > .ssh > web-5 > projects > work > main.go > ...

```go
package main

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

func work() {
    time.Sleep(time.Millisecond * 50)
    fmt.Println("done")
}

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()
}

```

```
PS C:\Users\admin\.ssh\web-5\projects\work> go run main.go
done
done
done
done
done
done
done
done
done
done
PS C:\Users\admin\.ssh\web-5\projects\work>
```

Задание 2

Код main.go

main.go

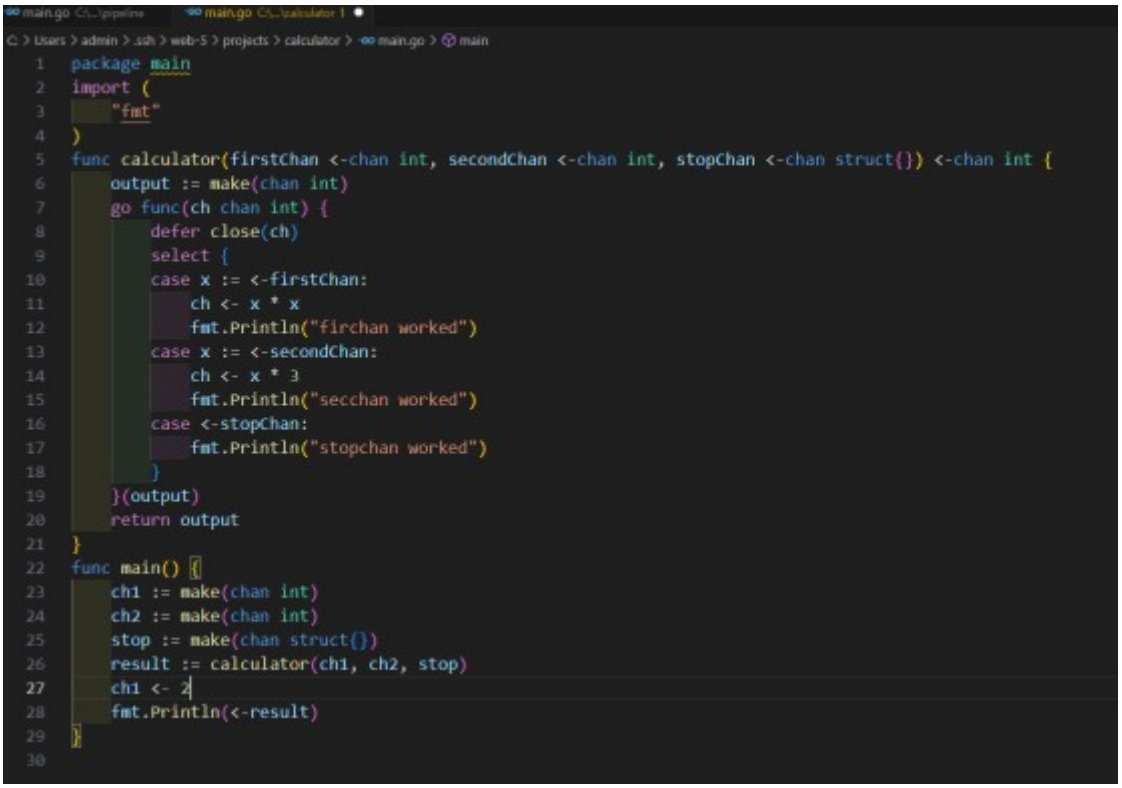
C: > Users > admin > .ssh > web-5 > projects > pipeline > main.go > ...

```go
package main

import "fmt"

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

func main() {
    inputStream := make(chan string)
    outputStream := make(chan string)
    go removeDuplicates(inputStream, outputStream)
    var str string
    fmt.Print("Text has been captured: ")
    fmt.Scanln(&str)
    go func() {
        defer close(inputStream)
        for _, st := range str {
            inputStream <- string(st)
        }
    }()
    fmt.Print("Result: ")
    for st := range outputStream {
        fmt.Print(st)
    }
    fmt.Println("\n", "End")
}
```

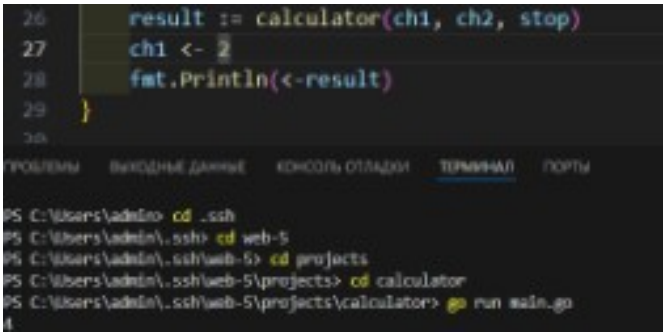
ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
PS C:\Users\admin> cd .ssh
PS C:\Users\admin\.ssh> cd web-5
PS C:\Users\admin\.ssh\web-5> cd projects
PS C:\Users\admin\.ssh\web-5\projects> cd pipeline
PS C:\Users\admin\.ssh\web-5\projects\pipeline> go run main.go
Text has been captured: 122555327
Result: 125327
 End
PS C:\Users\admin\.ssh\web-5\projects\pipeline>
```

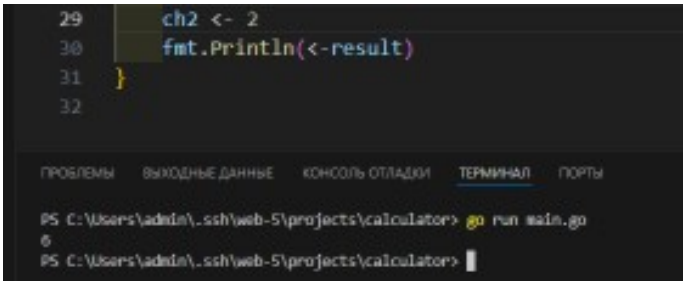
Задание 3
Код main.go



```go
package main
import (
    "fmt"
)
func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
    output := make(chan int)
    go func(ch chan int) {
        defer close(ch)
        select {
        case x := <-firstChan:
            ch <- x * x
            fmt.Println("firchan worked")
        case x := <-secondChan:
            ch <- x * 3
            fmt.Println("secchan worked")
        case <-stopChan:
            fmt.Println("stopchan worked")
        }
    }(output)
    return output
}
func main() {
    ch1 := make(chan int)
    ch2 := make(chan int)
    stop := make(chan struct{})
    result := calculator(ch1, ch2, stop)
    ch1 <- 2
    fmt.Println(<-result)
}
```



```
    result := calculator(ch1, ch2, stop)
    ch1 <- 2
    fmt.Println(<-result)
}

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

PS C:\Users\admin> cd .ssh
PS C:\Users\admin\.ssh> cd web-5
PS C:\Users\admin\.ssh\web-5> cd projects
PS C:\Users\admin\.ssh\web-5\projects> cd calculator
PS C:\Users\admin\.ssh\web-5\projects\calculator> go run main.go
4
```



```
    ch2 <- 2
    fmt.Println(<-result)
}

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

PS C:\Users\admin\.ssh\web-5\projects\calculator> go run main.go
6
PS C:\Users\admin\.ssh\web-5\projects\calculator>
```

Заключение: я научился основам асинхронного программирования на Golang.

Список использованных источников:

1. https://stepik.org/lesson/345547/