



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	<u>ИУ6-32Б</u> (Группа)	<u>06.10.2024</u> (Подпись, дата)	<u>Г.Д. Юдаков</u> (И.О. Фамилия)
Преподаватель		<u>08.10.2024</u> (Подпись, дата)	<u>В.Д. Шульман</u> (И.О. Фамилия)

Москва, 2024

Цель работы: изучение основ асинхронного программирования с использованием языка Golang

Ход работы.

1. Ознакомились с курсом <https://stepik.org/course/54403/info>
2. Сделали форк данного репозитория в GitHub, клонировали получившуюся копию локально, создали от мастера ветку дев и переключились на нее:

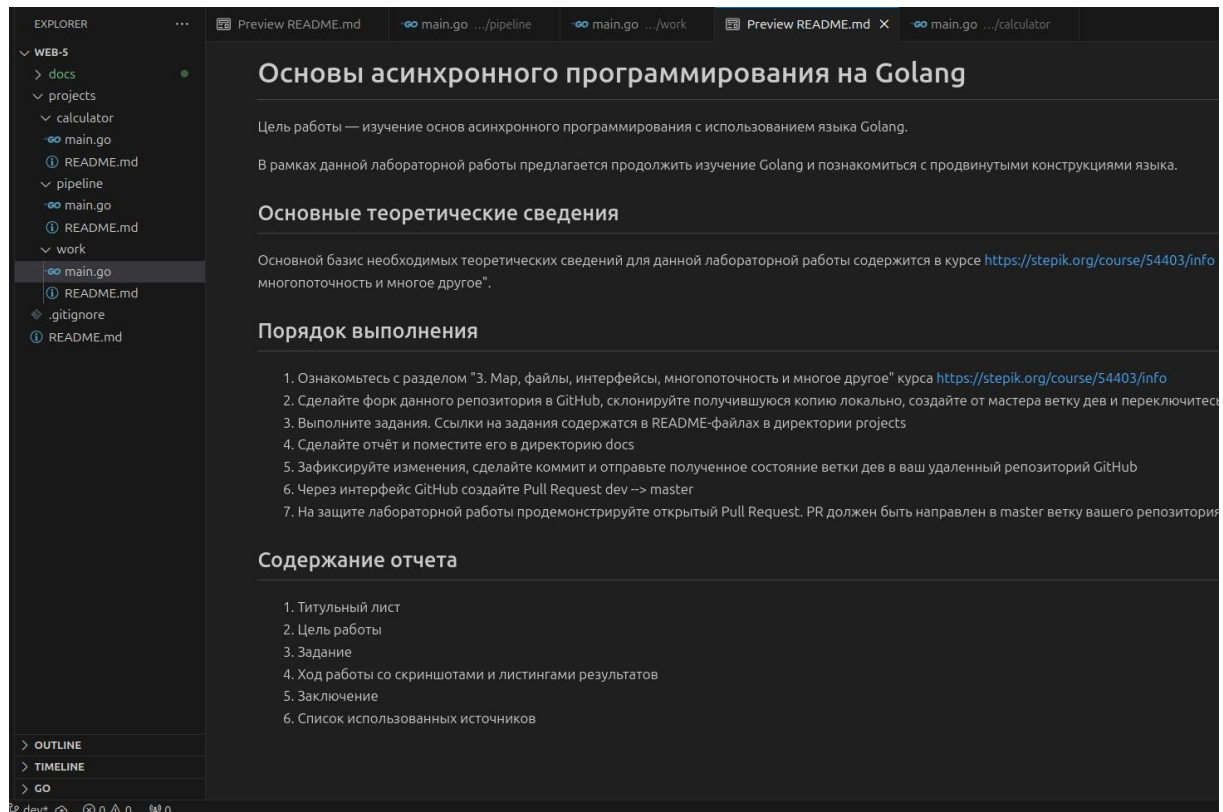


Рисунок 1 — Скопированный репозиторий

3. Решили 3 задачи на языке GoLang. Коды задач и результаты их работы прикрепили ниже:

**Задача 1**(Calculator) (Рис 2):

```
projects > calculator > -go main.go > ...

3  import (
4      |   "fmt"
5  )
6
7  func calculator(firstChan <-chan int, secondChan <-chan int,
8      resultChan := make(chan int)
9      go func() {
10         defer close(resultChan)
11         select {
12             case val := <-firstChan:
13                 |   resultChan <- val * val
14             case val := <-secondChan:
15                 |   resultChan <- val * 3
16             case <-stopChan:
17                 |   return
18         }
19     }()
20     return resultChan
21 }
22 |   ValeryBMSTU, 3 weeks ago • Initial commit
23 func main() {
24     firstChan := make(chan int)
25     secondChan := make(chan int)
26     stopChan := make(chan struct{})
27
28     result := calculator(firstChan, secondChan, stopChan)
29     go func() {
30         |   firstChan <- 7
31         |   secondChan <- 8
32         |   close(firstChan)
33         |   close(secondChan)
34         |   close(stopChan)
35     }()
36     for rest := range result {
37         |   fmt.Println(rest)
38     }
39
40 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... bash - calculator + v

- men-fish@Men-fish:~/web-5\$ cd /home/men-fish/web-5/projects/calculator
- men-fish@Men-fish:~/web-5/projects/calculator\$ go run main.go  
24
- men-fish@Men-fish:~/web-5/projects/calculator\$ go run main.go  
24
- men-fish@Men-fish:~/web-5/projects/calculator\$ go run main.go  
49
- men-fish@Men-fish:~/web-5/projects/calculator\$

Рисунок 2 — Задача 1 и ее вывод

**Задача 2**(Pipeline) (Рис 3):

```

1 package main
2
3 import "fmt"
4
5 func removeDuplicates(inputStream <-chan string, outputStream chan<- string) {
6     defer close(outputStream)
7     var prev string
8     for first := true; ; first = false {
9         if val, ok := <-inputStream; !ok {
10             return
11         } else if first || val != prev {
12             outputStream <- val
13             prev = val
14         }
15     }
16 }
17
18 func main() {
19     inputStream := make(chan string)
20     outputStream := make(chan string)
21
22     go removeDuplicates(inputStream, outputStream)
23
24     go func() {
25         for _, val := range []string{"a", "яяяяяяя", "b", "b", "a", "c", "c", "c"}
26         {
27             inputStream <- val
28         }
29         close(inputStream)
30     }()
31
32     for val := range outputStream {
33         fmt.Println(val)
34     }
35 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS bash - pipeline + v □

```

● men-fish@Men-fish:~/web-5$ cd /home/men-fish/web-5/projects/pipeline
● men-fish@Men-fish:~/web-5/projects/pipeline$ go run main.go
a
яяяяяяя
b
a
c
○ men-fish@Men-fish:~/web-5/projects/pipeline$

```

Рисунок 3 —Задача 2 и ее вывод

Задача 3(Work) (Рис 4):

```

1  package main
2
3  import (
4      "fmt"
5      "sync"
6  )
7
8  func work() {
9      fmt.Println("Work is done")
10 }
11
12 func main() {
13     var wg sync.WaitGroup
14
15     for i := 0; i < 10; i++ {
16         wg.Add(1)
17
18         go func() {
19             defer wg.Done()
20             work()
21         }()
22
23         wg.Wait()
24     }
25     fmt.Println("All work is done")
26 }
27
28

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... bash - work + v [ ] [ ]

```

• men-fish@Men-fish:~/web-5/projects/pipeline$ cd /home/men-fish/web-5/project
• men-fish@Men-fish:~/web-5/projects/work$ 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
All work is done
○ men-fish@Men-fish:~/web-5/projects/work$ [ ]

```

Рисунок 4 — Задача 3 и ее вывод

4. Зафиксировали изменения, сделали коммит и отправили полученное состояние ветки dev в удаленный репозиторий GitHub. Через интерфейс GitHub создали Pull Request dev --> master

Заключение: в ходе лабораторной работы изучили основы асинхронного программирования с использованием языка Golang. Освоили управление потоками и решили 3 задачи на эту тему.