



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. Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе №5

Название: Основы асинхронного программирования на Golang

Дисциплина: Основы WEB-разработки.

Студент

ИУ6-33Б

(Группа)

(Подпись, дата)

Е.В. Гредягина

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д.Шульман

(И.О. Фамилия)

Москва, 2024

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

2. Задание:

- Задание 1 pipeline – 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. Не забудьте закрыть канал ;) Функция должна называться `removeDuplicates()`

- Задание 2 work – 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. Пакет "sync" уже импортирован.

- Задание 3 calculator– 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.

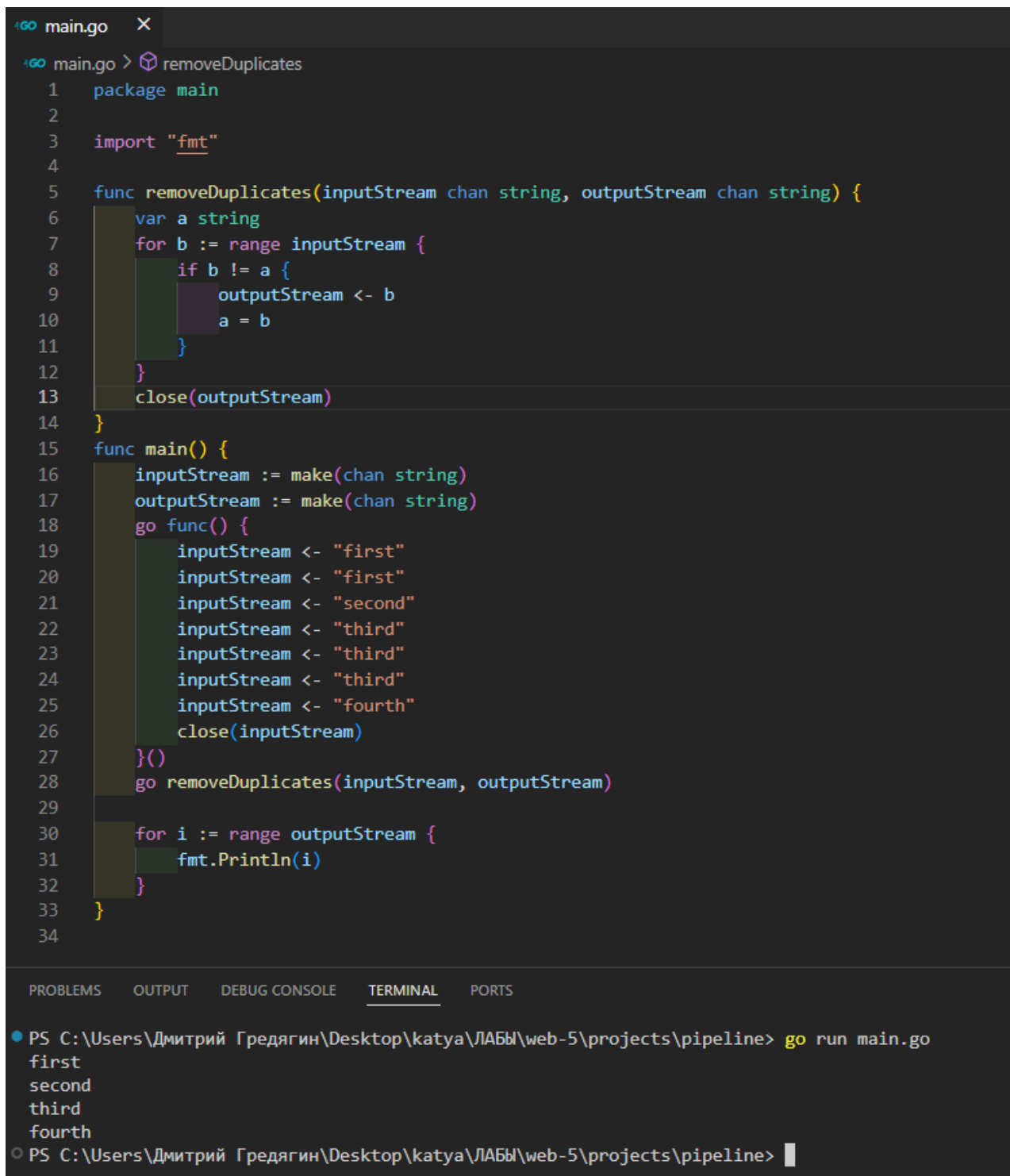
- в случае, если аргумент будет получен из канала `stopChan`, нужно просто завершить работу функции.

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.

3. Ход работы:

1. Ознакомьтесь с разделом "3. Map, файлы, интерфейсы, многопоточность и многое другое" курса <https://stepik.org/course/54403/info>
2. Сделали форк данного репозитория в GitHub, клонировали получившуюся копию локально, создали от мастера ветку `dev` и переключитесь на нее
3. Выполнили задания.

Коды сохранили в папке projects в подпапке с соответствующим названием задания. Скрины выполнения представлены ниже:



```
main.go X
main.go > removeDuplicates
1  package main
2
3  import "fmt"
4
5  func removeDuplicates(inputStream chan string, outputStream chan string) {
6      var a string
7      for b := range inputStream {
8          if b != a {
9              outputStream <- b
10             a = b
11         }
12     }
13     close(outputStream)
14 }
15 func main() {
16     inputStream := make(chan string)
17     outputStream := make(chan string)
18     go func() {
19         inputStream <- "first"
20         inputStream <- "first"
21         inputStream <- "second"
22         inputStream <- "third"
23         inputStream <- "third"
24         inputStream <- "third"
25         inputStream <- "fourth"
26         close(inputStream)
27     }()
28     go removeDuplicates(inputStream, outputStream)
29
30     for i := range outputStream {
31         fmt.Println(i)
32     }
33 }
34

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\pipeline> go run main.go
first
second
third
fourth
○ PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\pipeline> 
```

Рис. 1. Задача 1

```
main.go X
main.go > ...
2
3 import (
4     "fmt"
5     "sync"
6     "time"
7 )
8
9 func work() {
10     time.Sleep(time.Millisecond * 50)
11     fmt.Println("done")
12 }
13
14 func main() {
15     wg := new(sync.WaitGroup)
16     for i := 0; i < 10; i++ {
17         wg.Add(1)
18         go func() {
19             work()
20             wg.Done()
21         }()
22     }
23
24     wg.Wait()
25 }
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\work> go run main.go
done
done
done
done
done
done
done
done
done
done
done
- PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\work>

Рис. 2. Задача 2

```
main.go X
main.go > main
1 package main
2
3 import "fmt"
4
5 // реализовать calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int
6 func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
7     output := make(chan int)
8     go func() {
9         defer close(output)
10        select {
11            case a := <-firstChan:
12                output <- a * a
13            case b := <-secondChan:
14                output <- b * 3
15            case <-stopChan:
16                return
17        }
18    }
19    return output
20 }
21
22
23 func main() {
24     ch1, ch2 := make(chan int), make(chan int)
25     stop := make(chan struct{})
26
27     r := calculator(ch1, ch2, stop)
28     ch1 <- 4
29     //ch2 <- 3
30     close(stop)
31     fmt.Println(<-r)
32 }
33
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\calculator> go run main.go
- PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\calculator> 16

Рис. 3. Задача 3 (первый канал)

```
main.go X
main.go > ...
1  package main
2
3  import "fmt"
4
5  // реализовать calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int
6  func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
7      output := make(chan int)
8      go func() {
9          defer close(output)
10         select {
11             case a := <-firstChan:
12                 output <- a * a
13             case b := <-secondChan:
14                 output <- b * 3
15             case <-stopChan:
16                 return
17         }
18     }()
19     return output
20 }
21
22
23 func main() {
24     ch1, ch2 := make(chan int), make(chan int)
25     stop := make(chan struct{})
26
27     r := calculator(ch1, ch2, stop)
28     //ch1 <- 4
29     ch2 <- 3
30     close(stop)
31     fmt.Println(<-r)
32 }
33
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\calculator> go run main.go
9
PS C:\Users\Дмитрий Гредягин\Desktop\katya\ЛАБЫ\web-5\projects\calculator>
```

Рис. 4. Задача 3 (второй канал)

4. Сделали отчёт и поместите его в директорию docs
5. Зафиксировали изменения, сделали коммит и отправили полученное состояние ветки дев в удаленный репозиторий GitHub
6. Через интерфейс GitHub создали Pull Request dev --> master

4. **Заключение:** в ходе лабораторной работы я изучила основы асинхронного программирования с использованием языка Golang.

5. Список используемых источников:

<https://stepik.org/course/54403/info> (курс на Степике)