



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: Основы web программирования

Студент

ИУ6-32Б

(Группа)

(Подпись, дата)

И.В.Порохницкий

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д.Шульман

(И.О. Фамилия)

Москва, 2024

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

Задание

1. Ознакомьтесь с разделом "3. Map, файлы, интерфейсы, многопоточность и многое другое" курса <https://stepik.org/course/54403/info>
2. Сделайте форк данного репозитория в GitHub, склонируйте получившуюся копию локально, создайте от мастера ветку dev и переключитесь на неё
3. Выполните задания. Ссылки на задания содержатся в README-файлах в директории projects
4. Сделайте отчёт и поместите его в директорию docs
5. Зафиксируйте изменения, сделайте коммит и отправьте полученное состояние ветки dev в ваш удаленный репозиторий GitHub
6. Через интерфейс GitHub создайте Pull Request dev --> master

Ход работы

1. Ознакомился с курсом
2. Сделал форк данного репозитория в GitHub, клонировал получившуюся копию локально, создал от мастера ветку dev и переключился на нее
3. Выполнил 3 задания:

Задание №1 “calculator (Рис. 1):

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.

```
calculator > -go main.go > main
1  package main
2
3  import (
4      "fmt"
5  )
6  func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
7      resChan := make(chan int)
8      go func(){
9          defer close(resChan)
10         select{
11             case val, _ := <-firstChan:
12                 resChan <- val * val
13             case val, _ := <-secondChan:
14                 resChan <- val * 3
15             case <-stopChan:
16                 return
17         }
18     }()
19     return resChan
20 }
21
22 func main() {
23     firstChan := make(chan int)
24     secondChan := make(chan int)
25     stopChan := make(chan struct{})
26     resChan := calculator(firstChan, secondChan, stopChan)
27
28     go func() {
29         firstChan <- 4
30         close(firstChan)
31     }()
32     go func() {
33         secondChan <- 5
34         close(secondChan)
35         close(stopChan)
36     }()
37
38     for res := range resChan{
39         fmt.Println(res)
40     }
41
42 }
43
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Done] exited with code=0 in 0.597 seconds

[Running] go run "c:\Users\asset\web-5\projects\calculator\main.go"

16

[Done] exited with code=0 in 0.594 seconds

[Running] go run "c:\Users\asset\web-5\projects\calculator\main.go"

15

Рис.1

Задание №2 “pipeline”(Рис. 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.

```
pipeline > -go main.go > ...
```

```
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
9      for value := range inputStream {
10         if value != prev {
11             outputStream <- value
12             prev = value
13         }
14     }
15 }
16
17 func main() {
18     var strIn string
19     fmt.Scan(&strIn)
20     in := make(chan string)
21     out := make(chan string)
22     go removeDuplicates(in, out)
23     go func () {
24         for _, val:= range strIn{
25             in <- string(val)
26         }
27         close(in)
28     }()
29
30     for val := range out{
31         fmt.Print(val)
32     }
33
34 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\asset\web-5\projects>
```

```
cd pipeline
```

```
PS C:\Users\asset\web-5\projects\pipeline> go run main.go
```

```
112223443322
```

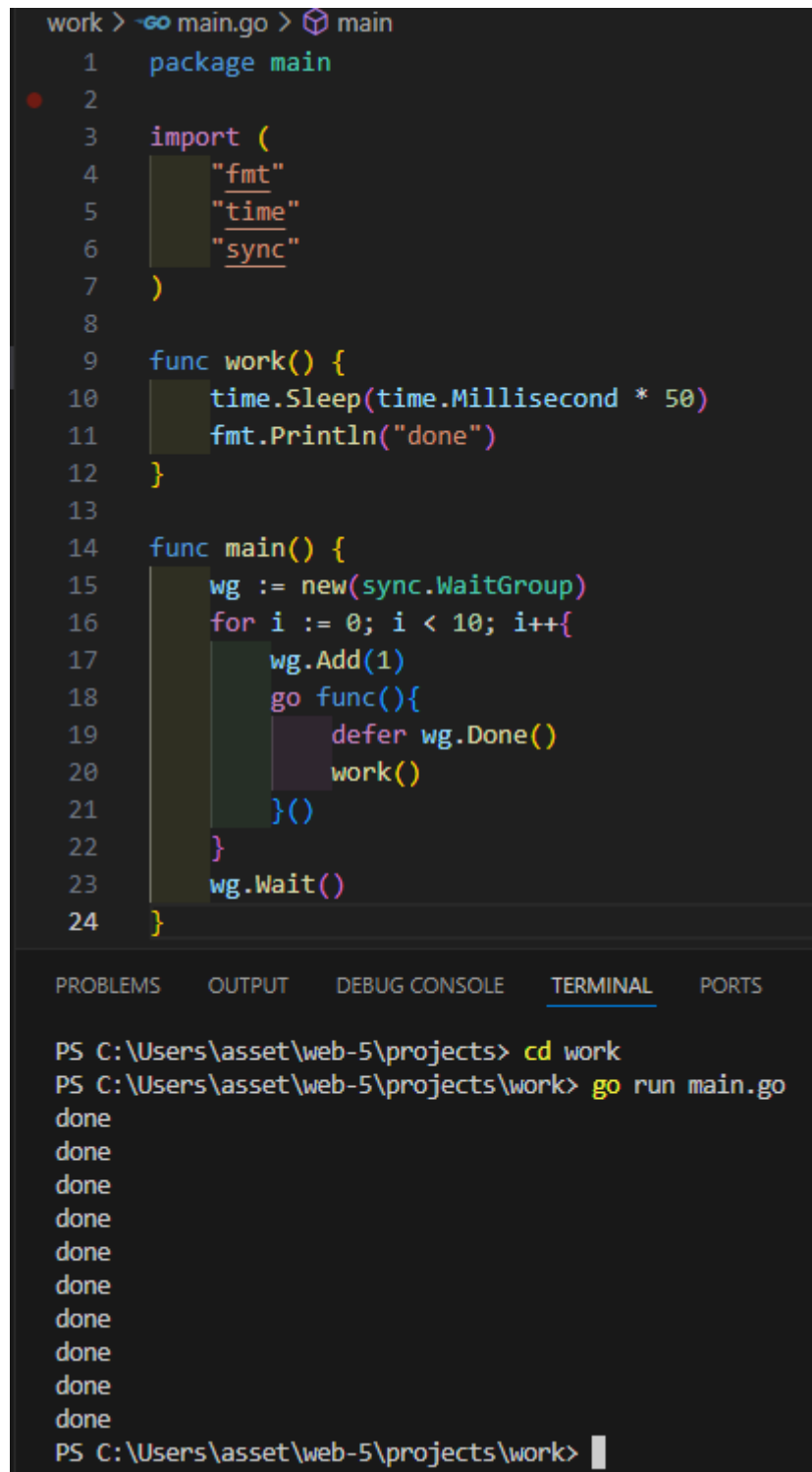
```
123432
```

```
PS C:\Users\asset\web-5\projects\pipeline> █
```

Рис.2

Задание №3 “work”(Рис. 3):

вам необходимо в отдельных горутинах вызвать функцию `work()` 10 раз и дождаться результатов выполнения вызванных функций.



```
work > -go main.go > main
1  package main
2
3  import (
4      "fmt"
5      "time"
6      "sync"
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             defer wg.Done()
20             work()
21         }()
22     }
23     wg.Wait()
24 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

рис. 3

4. Сделал отчёт и поместил его в директорию docs
Зафиксировал изменения, сделал коммит и отправил полученное
состояние ветки дев в удаленный репозиторий GitHub

Вывод

Я изучил основы асинхронного программирования с использованием языка Golang.