



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: Языки интернет-программирования

Студент

ИУ6-31Б

(Группа)

(Подпись, дата)

Л.В. Зимин

(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".
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
7. На защите лабораторной работы продемонстрируйте открытый Pull Request. PR должен быть направлен в master ветку вашего репозитория.

Ход работы:

1. Задача «Calculator»

Условие:

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.

Рисунок 1 — Условие задачи №1

Решение:

```
package main
```

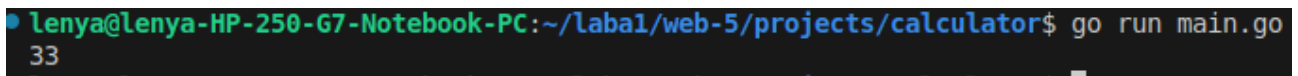
```
import (  
    "fmt"  
    "sync"  
)
```

```
func main() {  
    firstch := make(chan int)  
    secondch := make(chan int)  
    stopch := make(chan struct{})  
    ch := calculator(firstch, secondch, stopch)  
    wg := sync.WaitGroup{}  
    wg.Add(1)  
    go func() {  
        for ans := range ch {  
            fmt.Println(ans)  
        }  
        wg.Done()  
    }()  
    secondch <- 11  
    wg.Wait()
```

```
}
```

```
func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan
struct{}) <-chan int {
    res := make(chan int)
    go func(res chan int) {
        defer close(res)
        select {
            case x := <-firstChan:
                res <- x * x
            case x := <-secondChan:
                res <- x * 3
            case <-stopChan:
            }
        }(res)
    }
    return res
}
```

Тестирование:



```
lenya@lenya-HP-250-G7-Notebook-PC:~/lab1/web-5/projects/calculator$ go run main.go
33
```

Рисунок 2 — Тестирование задачи №1

2. Задача «Pipeline»

Условие:

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()`

Выводить или вводить ничего не нужно!

Рисунок 3 — Условие задачи №2

Решение

```
package main
```

```
import "fmt"
```

```
func main() {
```

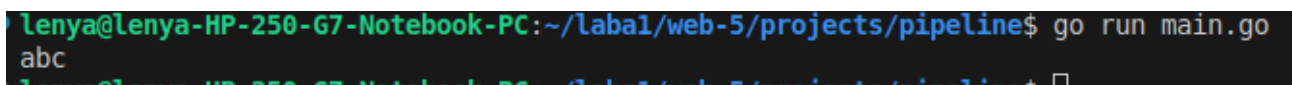
```

inputStream := make(chan string)
outputStream := make(chan string)
go removeDuplicates(inputStream, outputStream)
go func() {
    inputStream <- "a"
    inputStream <- "a"
    inputStream <- "b"
    inputStream <- "b"
    inputStream <- "c"
    close(inputStream)
}()
for x := range outputStream {
    fmt.Print(x)
}
fmt.Print("\n")
}

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

```

Тестирование:



```

lenya@lenya-HP-250-G7-Notebook-PC:~/lab1/web-5/projects/pipeline$ go run main.go
abc
lenya@lenya-HP-250-G7-Notebook-PC:~/lab1/web-5/projects/pipeline$

```

Рисунок 4 -Тестирование задачи №2

3. Задача «Work»

Условие:

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.

Рисунок 5 — Условие задачи №3

Решение:

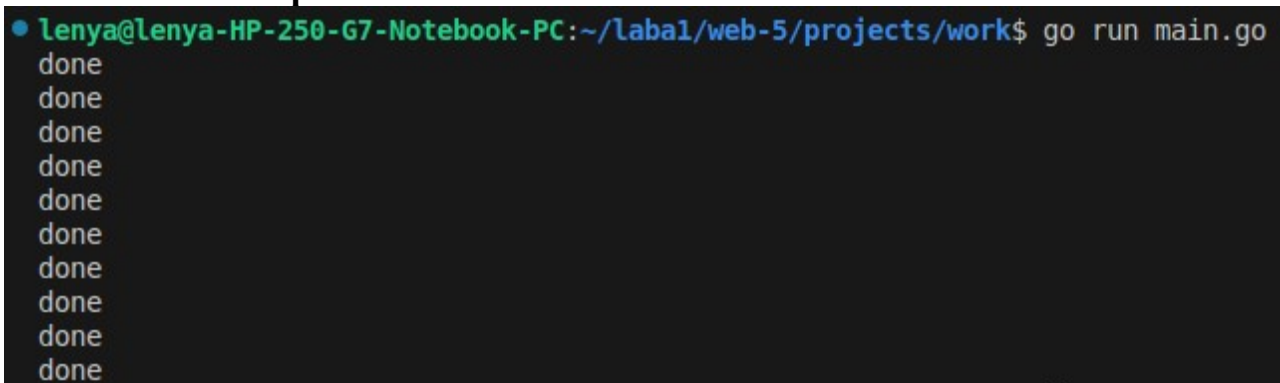
```
package main

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()
}
```

Тестирование:



```
● lenya@lenya-HP-250-G7-Notebook-PC:~/lab1/web-5/projects/work$ go run main.go
done
done
done
done
done
done
done
done
done
done
done
```

Рисунок 6 — Тестирование задачи №3

Вывод: асинхронность Golang помогает выполнять параллельные задачи, а также вычислять результат для входного потока данных.

Список источников:

- Сайт: <https://stepik.org/>