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) К.С. Доронина (И.О. Фамилия)

Преподаватель (Подпись, дата) В.Д. Шульман (И.О. Фамилия)

Москва, 2024

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

Порядок выполнения:

1. Ознакомиться с разделом 3 курса Stepik
2. Сделать форк репозитория с лабораторной работой
3. Выполнить задания в директории projects
4. Сделать отчет
5. Зафиксировать изменения и отправить изменения
6. Сделать Pull Request

Выполнение:

Задание 1 – pipeline (рис. 1)

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 1 – текст задания pipeline

Выполненное задание (рис. 2,3)

main.go > removeDuplicates

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

func main() {
    inputStream := make(chan string)
    outputStream := make(chan string)
    go removeDuplicates(inputStream, outputStream)

    var input string
    fmt.Printf("Введите строку с дубликатами: ")
    fmt.Scanln(&input)

    go func() {
        defer close(inputStream)

        for _, v := range input {
            inputStream <- string(v)
        }
    }()

    fmt.Printf("Строка без дубликатов: ")
    for v := range outputStream {
        fmt.Printf("%s", v)
    }
```

Рисунок 2 – код програмы задание 1

```
(base) kristinadoronina@MacBook-Pro-Kristina pipeline % go run main.go
Введите строку с дубликатами: 1122334455
Строка без дубликатов: 12345
(base) kristinadoronina@MacBook-Pro-Kristina pipeline % go run main.go
Введите строку с дубликатами: 9887665
Строка без дубликатов: 98765
(base) kristinadoronina@MacBook-Pro-Kristina pipeline % go run main.go
Введите строку с дубликатами: tonnight
Строка без дубликатов: tonight
(base) kristinadoronina@MacBook-Pro-Kristina pipeline %
```

Рисунок 3 – пример выполнения

Задание 2 – work (рис. 4)

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 4 – текст задания work

Выполнение задания (рис. 5,6)

```
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      var wg sync.WaitGroup
16
17      for i := 0; i < 10; i++ {
18          wg.Add(1)
19          go func() {
20              defer wg.Done()
21              work()
22          }()
23      }
24      wg.Wait()
25  }
```

Рисунок 5 – код программы

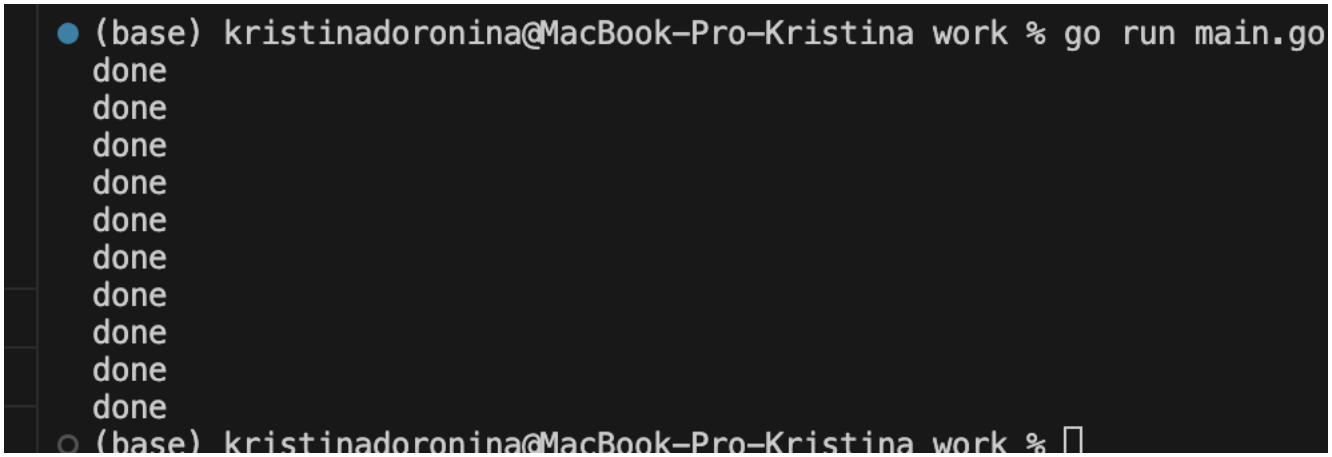


Рисунок 6 – пример выполнения

Задание 3 – calculator (рис. 7)

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.

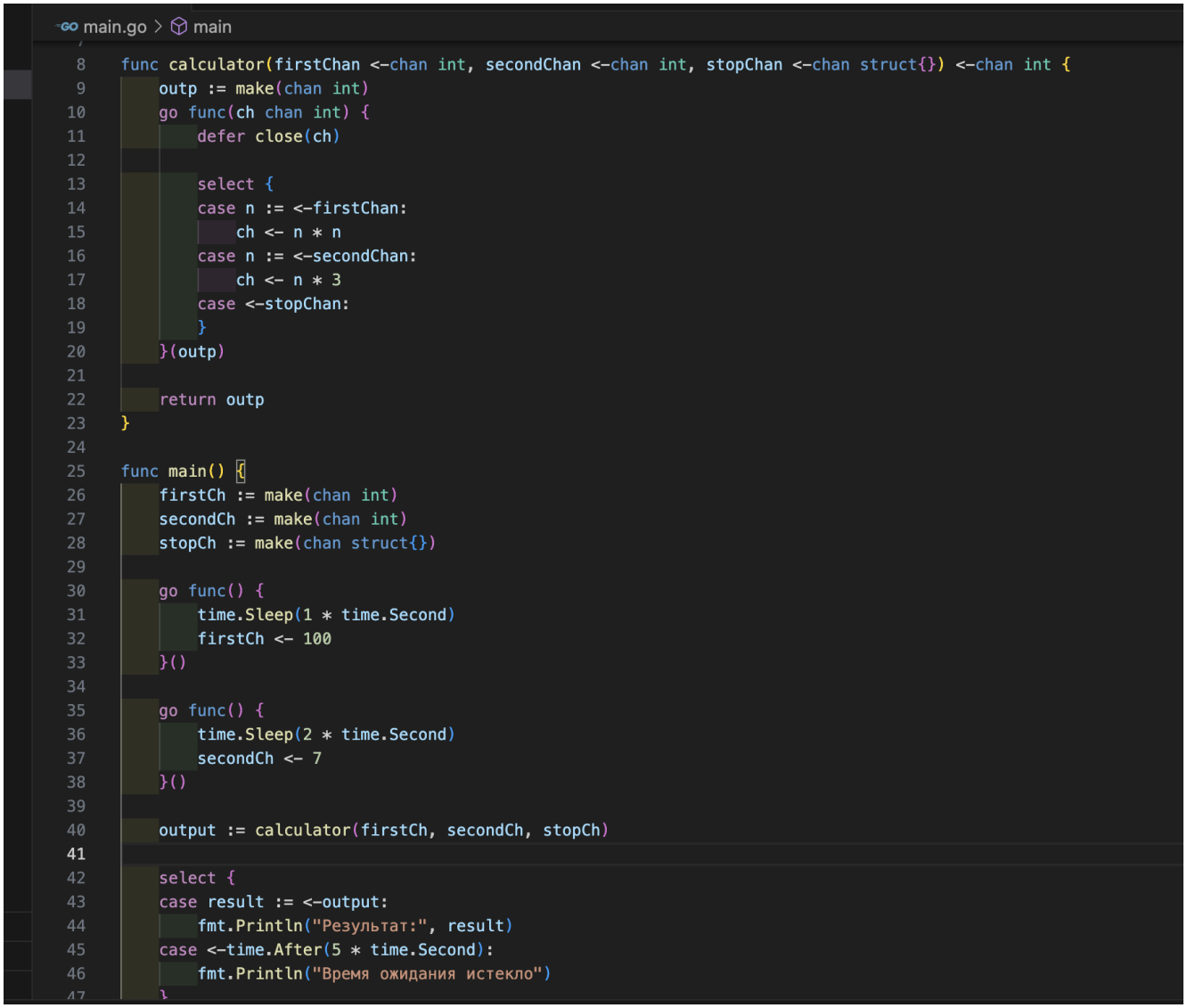
- в случае, если аргумент будет получен из канала firstChan, в выходной (возвращенный) канал вы должны отправить квадрат аргумента.
- в случае, если аргумент будет получен из канала secondChan, в выходной (возвращенный) канал вы должны отправить результат умножения аргумента на 3.
- в случае, если аргумент 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.

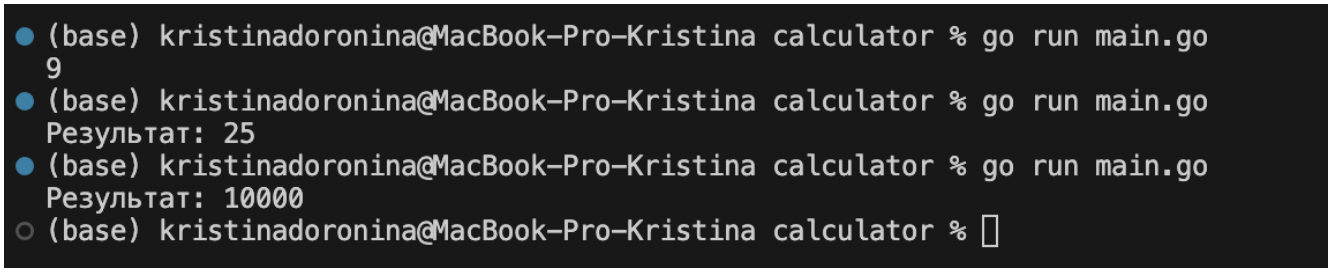
Рисунок 7 – текст задания

Выполнение задания (рис. 8,9)



```go
func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
    outp := make(chan int)
    go func(ch chan int) {
        defer close(ch)

        select {
        case n := <-firstChan:
            ch <- n * n
        case n := <-secondChan:
            ch <- n * 3
        case <-stopChan:
        }
    }(outp)

    return outp
}

func main() {
    firstCh := make(chan int)
    secondCh := make(chan int)
    stopCh := make(chan struct{})

    go func() {
        time.Sleep(1 * time.Second)
        firstCh <- 100
    }()

    go func() {
        time.Sleep(2 * time.Second)
        secondCh <- 7
    }()

    output := calculator(firstCh, secondCh, stopCh)

    select {
    case result := <-output:
        fmt.Println("Результат:", result)
    case <-time.After(5 * time.Second):
        fmt.Println("Время ожидания истекло")
    }
```

Рисунок 8 – код программы



```
(base) kristinadoronina@MacBook-Pro-Kristina calculator % go run main.go
9
(base) kristinadoronina@MacBook-Pro-Kristina calculator % go run main.go
Результат: 25
(base) kristinadoronina@MacBook-Pro-Kristina calculator % go run main.go
Результат: 10000
(base) kristinadoronina@MacBook-Pro-Kristina calculator %
```

Рисунок 9 – пример выполнения

Заключение:

Выполнили задания связанные с асинхронным программированием на Golang.