|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Отчет**

**по лабораторной работе № 5**

**Название:** Основы асинхронного программирования на Golang

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б |  |  | Д.Е. Горячев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

1. **Цель лабораторной работы**

Изучение основ асинхронного программирования с использованием языка Golang.

1. **Задание**

Продолжить изучение Golang и познакомиться с продвинутыми конструкциями языка.

1. **Ход работы**
2. Создание собственной копии репозитория с данной лабораторной работой, а также клонирования текущего репозитория на локальную машину – было сделано заранее.
3. Разработка calculator:

Задание:

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.

Код:

package main

import "fmt"

// реализовать calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int

func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {

    ch := make(chan int)

    var val int

    go func() {

        for {

            select {

            case val = <-firstChan:

                ch <- val \* val

            case val = <-secondChan:

                ch <- val \* 3

            case <-stopChan:

                close(ch)

                return

            }

        }

    }()

    return ch

}

func main() {

    // здесь должен быть код для проверки правильности работы функции calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int

    //type Str struct{}

    firstChan, secondChan := make(chan int), make(chan int)

    stopChan := make(chan struct{})

    resultChan := calculator(firstChan, secondChan, stopChan)

    firstChan <- 10

    fmt.Println(<-resultChan)

    secondChan <- 15

    fmt.Println(<-resultChan)

    close(stopChan)

    fmt.Println(<-resultChan)

}

Результат:

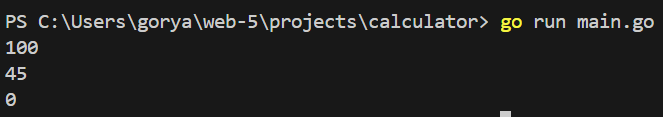


Рисунок 1 – ход выполнения программы

1. Разработка pipeline:

Задание:

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()

Код:

package main

import (

    "fmt"

)

func removeDuplicates(inputStream <-chan string, outputStream chan<- string) {

    var lastValue string

    for currentValue := range inputStream {

        if currentValue != lastValue {

            outputStream <- currentValue

            lastValue = currentValue

        }

    }

    close(outputStream)

}

func main() {

    inputStream := make(chan string)

    outputStream := make(chan string)

    go removeDuplicates(inputStream, outputStream)

    go func() {

        inputStream <- "hello"

        inputStream <- "hello"

        inputStream <- "world"

        inputStream <- "world"

        inputStream <- "!"

        inputStream <- "!"

        close(inputStream)

    }()

    for result := range outputStream {

        fmt.Println(result)

    }

}

Результат:

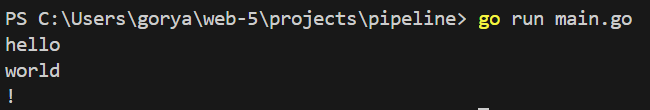


Рисунок 2 – результат работы программы

1. Разработка work:

Задание:

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.

Код:

package main

import (

    "fmt"

    "sync"

    "time"

)

func work() {

    time.Sleep(time.Millisecond \* 50)

    fmt.Println("done")

}

func main() {

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

    wg := new(sync.WaitGroup)

    for i := 0; i < 10; i++ {

        wg.Add(1)

        go func(wg \*sync.WaitGroup) {

            defer wg.Done()

            work()

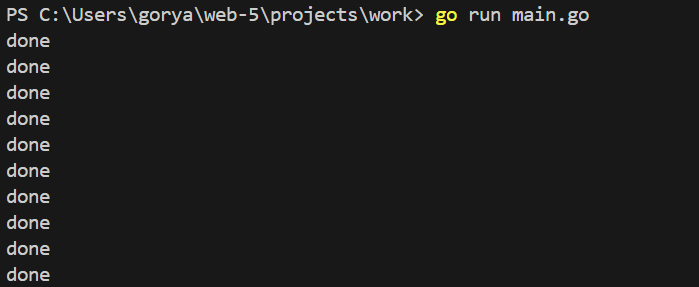
        }(wg)

    }

    wg.Wait()

}

Результат:



1. **Заключение**

Ознакомились с продвинутыми конструкциями языка Golang.

1. **Использованные источники**

<https://stepik.org/course/54403/info>

<https://github.com/ValeryBMSTU/web-5>