МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Домашнее задание на тему

«Разработка распределенного (параллельного) поисковика слов на языке Go»

Выполнила:
Студент группы ИУ5-36Б
Топорец Г.А.

Проверил:
Гапанюк Ю.Е.

Москва 2025

Задание:

Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).

Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.

Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.

В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).

В случае создания проекта необходимо детально комментировать код.

При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.

Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Мини-проект:

Язык программирования Go (Golang). Задача: Разработка программного средства для параллельного многопоточного поиска подстроки в массиве текстовых данных.Технические условия: Входные данные: Директория с набором текстовых файлов (20 - 30 штук). Использование примитивов синхронизации Go для параллельной обработки.

Описание реализации:

Для решения задачи выбран шаблон "Producer-Consumer" (Производитель-Потребитель).

Main: Сканирует файловую систему и инициализирует канал.

Goroutines: Каждый файл обрабатывается в отдельной goroutine (это параллельные операции в языке программирования Go, которые могут выполняться независимо от функции, в которой они запущены.). Для синхронизации завершения используется sync.WaitGroup.

Communication: Передача найденных совпадений в центральный поток вывода осуществляется через потокобезопасный канал (chan).

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "path/filepath"
    "strings"
    "sync"
)

// Структура для хранения метаданных найденного вхождения
type Match struct {
    FilePath string
    LineNum  int
    Content  string
}

func main() {
    const targetDir = "./logs"
    const query = "ERROR"

    files, err := os.ReadDir(targetDir)
    if err != nil {
       fmt.Fprintf(os.Stderr, "Ошибка доступа к директории: %v\n", err)
       os.Exit(1)
    }

    matchChan := make(chan Match) // Канал для результатов
    var wg sync.WaitGroup

    // Итерация по файлам и запуск горутин
    for _, f := range files {
       if f.IsDir() || !strings.HasSuffix(f.Name(), ".txt") {
          continue
       }

       wg.Add(1)
       go func(fileName string) {
          defer wg.Done()
```

```
            searchInFile(filepath.Join(targetDir, fileName), query,
matchChan)
        }(f.Name())
    }

    // Горутина-монитор для закрытия канала
    go func() {
        wg.Wait()
        close(matchChan)
    }()

    // Сбор результатов (Consumer)
    fmt.Printf("Результаты поиска [Запрос: %s]:\n", query)
    fmt.Println(strings.Repeat("-", 50))

    for m := range matchChan {
        fmt.Printf("%s:%d | %s\n", m.FilePath, m.LineNum, m.Content)
    }
}

func searchInFile(path string, query string, ch chan<- Match) {
    file, err := os.Open(path)
    if err != nil {
        return
    }
    defer file.Close()

    scanner := bufio.NewScanner(file)
    lineIdx := 1
    for scanner.Scan() {
        line := scanner.Text()
        if strings.Contains(line, query) {
            ch <- Match{
                FilePath: path,
                LineNum:  lineIdx,
                Content:  strings.TrimSpace(line),
            }
        }
        lineIdx++
    }
}
```

Тестирование:

```
API server listening at: 127.0.0.1:61739
Результаты поиска [Запрос: ERROR]:
---------------------------------------------------
logs\4.txt:3 | 192.168.1.10 POST /login 500 -- ERROR: Internal Server Error (Database unreachable)
logs\10.txt:1 | Lorem ipsum dolor sit amet, ERROR consectetur adipiscing elit. Sed do eiusmod tempor incid
logs\3.txt:3 | query: DROP TABLE production; -- ERROR: Permission denied for user 'guest'
logs\1.txt:3 | [10:00:10] ERROR: Failed to load driver 'nvidia-440'.
logs\7.txt:2 | ERROR: Sector 0 is corrupted on disk /dev/sdb.
logs\7.txt:3 | ERROR: Re-allocation failed.
logs\5.txt:3 | 2. Check why ERROR constant is not defined in the header file.
logs\9.txt:3 | ERROR: Handshake_failure (code 40) at 127.0.0.1.
logs\8.txt:5 | # log_level=ERROR
logs\7.txt:4 | ERROR: System shutdown initiated to prevent data loss.
logs\6.txt:5 | ERROR: standard_init_linux.go:228: exec user process caused "no such file or directory"
```