

**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе № 5  
«Работа с файлами в Go»

Выполнил:  
студент группы ИУ5-31Б  
Князев А.М.

Проверил:  
преподаватель каф. ИУ5  
Нардид А.Н.

Москва, 2024 г.

## Описание задания

### 1. Запись в файл:

- Создать текстовый файл и записать в него строку, переданную в функцию `writeToFile`.

### 2. Чтение из файла:

- Реализовать функцию для чтения содержимого текстового файла и возврата строки с его содержимым.

### 3. Подсчёт количества слов:

- Создать функцию, которая принимает текст в виде строки, разделяет его на слова и подсчитывает их количество.

### 4. Копирование файла:

- Реализовать функцию копирования содержимого одного файла в другой.

### 5. Обработка ошибок:

- Предусмотреть обработку возможных ошибок (например, при создании, чтении или копировании файла) с использованием конструкции `fmt.Errorf` для вывода подробных сообщений об ошибках.

### 6. Основной алгоритм работы программы:

- Записать произвольную строку в файл `output.txt`.
- Прочитать содержимое файла `output.txt`.
- Подсчитать количество слов в прочитанной строке.
- Скопировать файл `output.txt` в новый файл `copy.txt`.
- Вывести результаты выполнения всех этапов работы программы в консоль.

## Текст программы

Файл *lab4.go*

```
package main

import (
    "bufio"
    "fmt"
    "io"
    "os"
    "strings"
)

func main() {

    err := writeToFile("output.txt", "ПиКЯП 3 семестр Князев")
    if err != nil {
        fmt.Printf("Ошибка при записи в файл: %v\n", err)
        return
    }

    contents, err := readFile("output.txt")
    if err != nil {
        fmt.Printf("Ошибка при чтении файла: %v\n", err)
        return
    }
    fmt.Println("Текст из файла:", contents)

    wordCount := countWords(contents)
    fmt.Println("Количество слов:", wordCount)

    err = copyFile("output.txt", "copy.txt")
    if err != nil {
        fmt.Printf("Ошибка при копировании файла: %v\n", err)
        return
    }
    fmt.Println("Файл успешно скопирован.")
}

func writeToFile(filename, text string) error {
    file, err := os.Create(filename)
    if err != nil {
        return fmt.Errorf("не удалось создать файл: %w", err)
    }
    defer file.Close()

    writer := bufio.NewWriter(file)
    _, err = writer.WriteString(text)
    if err != nil {
```

```

        return fmt.Errorf("не удалось записать текст в файл: %w", err)
    }
    return writer.Flush()
}

func readFile(filename string) (string, error) {
    data, err := os.ReadFile(filename)
    if err != nil {
        return "", fmt.Errorf("не удалось прочитать файл: %w", err)
    }
    return string(data), nil
}

func countWords(text string) int {
    return len(strings.Fields(text))
}

func copyFile(source, dest string) error {
    sourceFile, err := os.Open(source)
    if err != nil {
        return fmt.Errorf("не удалось открыть исходный файл: %w", err)
    }
    defer sourceFile.Close()

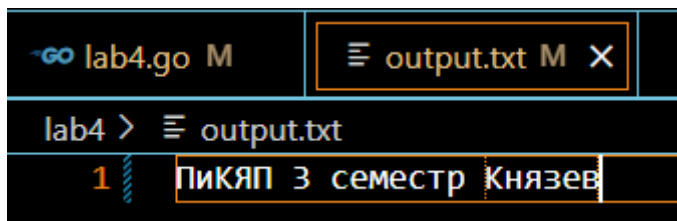
    destFile, err := os.Create(dest)
    if err != nil {
        return fmt.Errorf("не удалось создать целевой файл: %w", err)
    }
    defer destFile.Close()

    _, err = io.Copy(destFile, sourceFile)
    if err != nil {
        return fmt.Errorf("не удалось скопировать данные: %w", err)
    }
    return nil
}

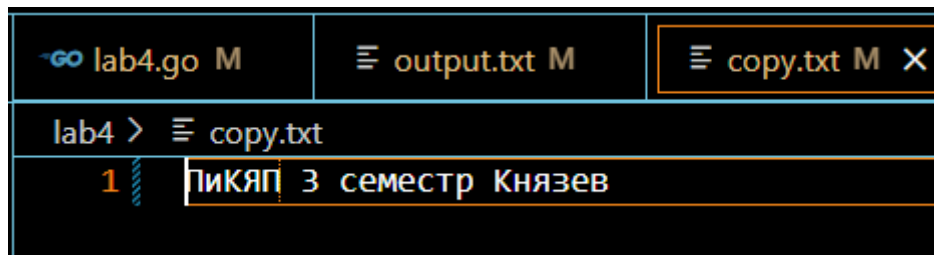
```

## Экранные формы с примерами выполнения программы

Файл *output.txt*



Файл *copy.txt*



The image shows a code editor interface with a dark background. At the top, there are three tabs: 'lab4.go M', 'output.txt M', and 'copy.txt M X'. The 'copy.txt' tab is active. Below the tabs, the text 'lab4 > ≡ copy.txt' is displayed. The main editing area shows a single line of text: '1 ПИКЯП 3 семестр Князев'. The line number '1' is on the left, and the text is highlighted with a yellow selection box.