**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 № 4

«Работа со структурами и их методами на языке Go»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Кузнецов А.Д. |  | Нардид А.Н. |
|  |  |  |

Москва, 2024 г.

**Описание задания**

Разработать программу на языке Go для управления данными о студентах в учебном заведении. Программа должна поддерживать базовые функции работы со списком студентов, включая добавление, удаление, обновление и сортировку по курсу.

**Функциональные требования:**

1. **Создание структуры студента**:  
   Определить структуру Student, которая содержит следующие поля:
   * Id — уникальный идентификатор студента;
   * FirstName — имя студента;
   * SecondName — фамилия студента;
   * Age — возраст студента;
   * Faculty — название факультета;
   * Course — номер курса.
2. **Функции программы**:
   * NewStudent() — конструктор для создания нового студента.
   * AddStudent() — добавление студента в список.
   * RemoveStudent() — удаление студента по его Id.
   * UpdateStudent() — обновление данных студента по его Id.
   * PrintAllStudents() — вывод всех студентов на экран.
   * SortStudentsByCourse() — сортировка списка студентов по их курсу.
3. **Логика программы**:
   * Создать и наполнить список несколькими студентами.
   * Вывести всех студентов на экран.
   * Обновить данные одного из студентов и отобразить обновлённый список.
   * Удалить студента по идентификатору и вывести список.
   * Отсортировать студентов по курсу и показать отсортированный список.

**Дополнительные требования**:

* Структурировать код с использованием функций.
* Обеспечить корректное выполнение всех операций со списком студентов.
* Использовать слайсы для хранения данных и итерационные алгоритмы для работы с ними.

**Текст программы**

Файл *source.go*

package main

import "fmt"

type Student struct {

    Id         int

    FirstName  string

    SecondName string

    Age        int

    Faculty    string

    Course     int

}

func NewStudent(id int, firstName, secondName string, age int, faculty string, course int) Student {

    return Student{

        Id:         id,

        FirstName:  firstName,

        SecondName: secondName,

        Age:        age,

        Faculty:    faculty,

        Course:     course,

    }

}

func AddStudent(students \*[]Student, student Student) {

    \*students = append(\*students, student)

}

func RemoveStudent(students \*[]Student, id int) {

    for i, student := range \*students {

        if student.Id == id {

            \*students = append((\*students)[:i], (\*students)[i+1:]...)

            return

        }

    }

}

func UpdateStudent(students \*[]Student, id int, updatedStudent Student) {

    for i, student := range \*students {

        if student.Id == id {

            (\*students)[i] = updatedStudent

            return

        }

    }

}

func PrintAllStudents(students []Student) {

    for i := 0; i < len(students); i++ {

        fmt.Printf("Id: %d\nFirst Name: %s\nSecond Name: %s\nAge: %d\nFaculty: %s\nCourse: %d\n", students[i].Id, students[i].FirstName, students[i].SecondName, students[i].Age, students[i].Faculty, students[i].Course)

        fmt.Println("--------------")

    }

}

func SortStudentsByCourse(students []Student) []Student {

    var Sorted []Student = students

    for i := 0; i < len(Sorted)-1; i++ {

        for j := i + 1; j < len(Sorted); j++ {

            if Sorted[i].Course > Sorted[j].Course {

                Sorted[i], Sorted[j] = Sorted[j], Sorted[i]

            }

        }

    }

    return Sorted

}

Файл *main.go*

package main

import "fmt"

func main() {

    var students []Student

    student1 := NewStudent(1, "Артем", "Кузнецов", 19, "ИУ5", 3)

    student2 := NewStudent(2, "Алексей", "Князев", 19, "ИУ5", 2)

    student3 := NewStudent(3, "Дмитрий", "Егоров", 19, "ИУ5", 2)

    AddStudent(&students, student1)

    AddStudent(&students, student2)

    AddStudent(&students, student3)

    fmt.Println("Все студенты:")

    PrintAllStudents(students)

    fmt.Printf("\n============================\n\n")

    fmt.Println("Обновим данные студента")

    updatedStudent := NewStudent(2, "Алексей", "Князев", 20, "ИУ5", 2)

    UpdateStudent(&students, 2, updatedStudent)

    PrintAllStudents(students)

    fmt.Printf("\n============================\n\n")

    fmt.Println("Удалим студента:")

    RemoveStudent(&students, 3)

    PrintAllStudents(students)

    fmt.Printf("\n============================\n\n")

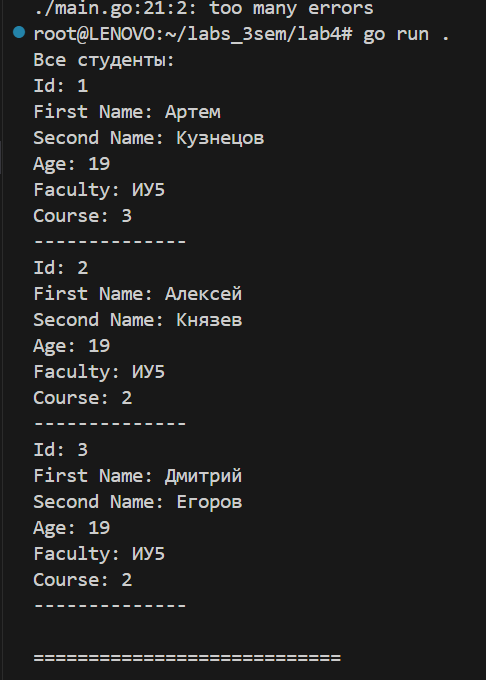
    fmt.Println("Отсортируем студентов по курсу:")

    sortedStudents := SortStudentsByCourse(students)

    PrintAllStudents(sortedStudents)

}

**Экранные формы с примерами выполнения программы**

****

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**