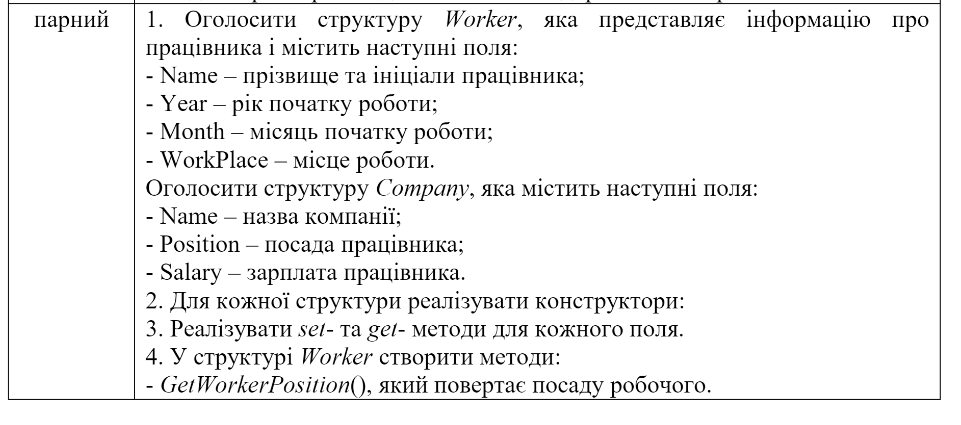
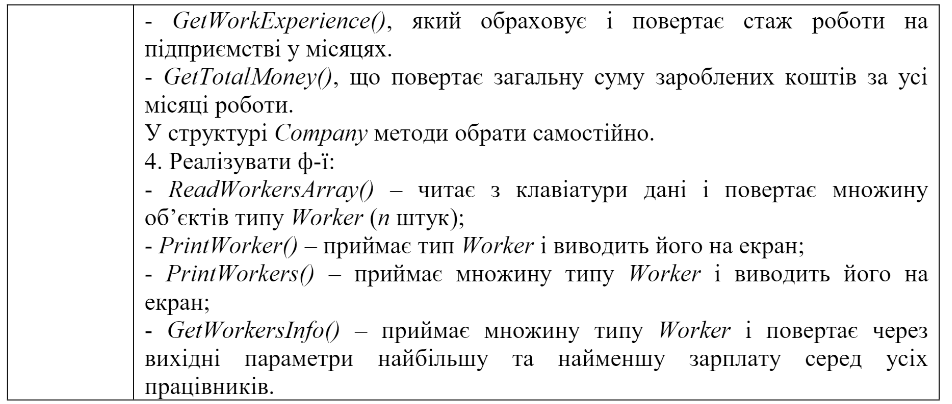
**ЛАБОРАТОРНА РОБОТА № 5**

**Управляючі конструкції в мові Go. Функції. Введення та виведення інформації у консоль**

***Мета:*** засвоїти принципи проектування та оголошення структур та інтерфейсів; вивчити особливості реалізації структур та їх методів.

**Хід роботи**





Структури Company та Worker:

type Company struct {

    Name     string *// назва компанії*

    Position string *// посада працівника*

    Salary   int    *// зарплата працівника*

}

type Worker struct {

    Name      string  *// прізвище та ініціали працівника*

    Year      int     *// рік початку роботи*

    Month     int     *// місяць початку роботи*

    WorkPlace Company *// місце роботи*

}

Конструктори структур:

func NewCompany(name, position string, salary int) \*Company {

*return* &Company{

        Name:     name,

        Position: position,

        Salary:   salary,

    }

}

func NewWorker(name string, year, month int, workPlace Company) \*Worker {

*return* &Worker{

        Name:      name,

        Year:      year,

        Month:     month,

        WorkPlace: workPlace,

    }

}

Set- та get- методи для кожного поля

*// set and get*

func (u \*Company) SetName(name string) {

    u.Name = name

}

func (u \*Company) GetName() string {

*return* u.Name

}

func (u \*Company) SetPosition(position string) {

    u.Position = position

}

func (u \*Company) GetPosition() string {

*return* u.Position

}

func (u \*Company) SetSalary(salary int) {

    u.Salary = salary

}

func (u \*Company) GetSalary() int {

*return* u.Salary

}

func (u \*Worker) SetName(name string) {

    u.Name = name

}

func (u \*Worker) GetName() string {

*return* u.Name

}

func (u \*Worker) SetYear(year int) {

    u.Year = year

}

func (u \*Worker) GetYear() int {

*return* u.Year

}

func (u \*Worker) SetMonth(month int) {

    u.Month = month

}

func (u \*Worker) GetMonth() int {

*return* u.Month

}

func (u \*Worker) SetWorkPlace(workPlace Company) {

    u.WorkPlace = workPlace

}

func (u \*Worker) GetWorkPlace() Company {

*return* u.WorkPlace

}

- *GetWorkerPosition*(), який повертає посаду робочого:

func (u \*Worker) GetWorkerPosition() string {

*return* u.WorkPlace.Position

}

func main() {

    workPlaceWorker := NewCompany("Tesla Motors", "Gen Director", 2000)

    worker := NewWorker("Kopylov V.", 2020, 4, \*workPlaceWorker)

    worker.Show()

    worker.SetName("Billy")

    worker.Show()

    workerPos := worker.GetWorkerPosition()

    fmt.Println("Посада ->", workerPos)



Рис.1. Результат GetWorkerPosition

- *GetWorkExperience()*, який обраховує і повертає стаж роботи на підприємстві у місяцях:

func (u \*Worker) GetWorkerExperience() int {

    t := time.Now()

    yearNow := t.Year()

    yearExp := u.Year - yearNow

    monthNow := t.Month()

    totalExp := int(math.Abs(float64(yearExp)))\*12 - 12 + (12 - u.Month) + int(monthNow)

*return* totalExp

}

experience := worker.GetWorkerExperience()

fmt.Println("Стаж роботи ->", experience, "місяців")



Рис.2. Результат GetWorkExperience

- *GetTotalMoney()*, що повертає загальну суму зароблених коштів за усі місяці роботи:

func (u \*Worker) GetTotalMoney() int {

    totalMonth := u.GetWorkerExperience()

    totalMoney := totalMonth \* u.WorkPlace.Salary

*return* totalMoney

}

money := worker.GetTotalMoney()

fmt.Println("Усього зароблених грошей за", experience, "місяців ->", money)



Рис.3. Результат GetTotalMoney

*CompanyShow()* – виводить на екран поля структури:

func (u \*Company) Show() {

    fmt.Println("Name ->", u.Name, "\nPosition ->", u.Position, "\nSalary ->", u.Salary)

}

workPlaceWorker.Show()



Рис.4. Результат CompanyShow

- *ReadWorkersArray()* – читає з клавіатури дані і повертає множину об’єктів типу *Worker* (*n* штук):

func ReadWorkersArray(numWorkers int) []Worker {

    workersArray := []Worker{}

*for* i := 0; i < numWorkers; i++ {

        var workerName, companyName, companyPosition string

        var workerYear, workerMonth, companySalary int

        fmt.Println("Введіть ПІБ працівника")

        fmt.Scanf("%s\n", &workerName)

        fmt.Println("Введіть рік початку роботи")

        fmt.Scanf("%d\n", &workerYear)

        fmt.Println("Введіть місяць початку роботи")

        fmt.Scanf("%d\n", &workerMonth)

        fmt.Println("Введіть назву компанії")

        fmt.Scanf("%s\n", &companyName)

        fmt.Println("Введіть посаду")

        fmt.Scanf("%s\n", &companyPosition)

        fmt.Println("Введіть заробітню плату")

        fmt.Scanf("%d\n", &companySalary)

        workPlaceWorker := NewCompany(companyName, companyPosition, companySalary)

        worker := NewWorker(workerName, workerYear, workerMonth, \*workPlaceWorker)*)*

        fmt.Println("Дані записані!")

        workersArray = append(workersArray, \*worker)

    }

*return* workersArray

}

*// -----------------------------------------*

    fmt.Println("==============================")

    var numWorkers int

    fmt.Printf("Кількість працівників -> ")

*for* {

        fmt.Scanf("%d\n", &numWorkers)

*if* numWorkers > 0 {

*break*

        }

    }

    workersArray := []Worker{}

    workersArray = ReadWorkersArray(numWorkers)

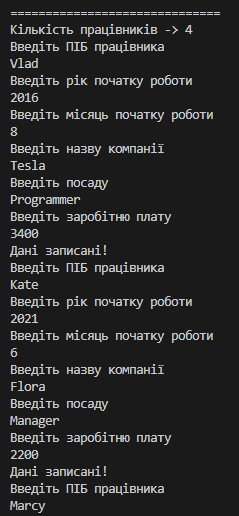


Рис.5. Результат ReadWorkersArray

- *PrintWorkers()* – приймає множину типу *Worker* і виводить його на екран:

func PrintWorkers(arr []Worker) {

*for* i, value := *range* arr {

        fmt.Println(i+1, "\nПІБ ->", value.Name, "\nМісце роботи ->", value.WorkPlace.Name, "\nПосада ->", value.WorkPlace.Position, "\nЗаробітня плата ->", value.WorkPlace.Salary)

    }

}

PrintWorkers(workersArray)

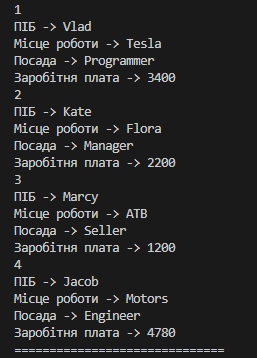


Рис.6. Результат PrintWorkers

- *PrintWorker()* – приймає тип *Worker* і виводить його на екран:

func PrintWorker(u Worker) {

    fmt.Println("Name ->", u.Name, "\nYear ->", u.Year, "\nMonth ->", u.Month, "\nWorkPlace ->", u.WorkPlace)

}

    PrintWorker(\*worker)

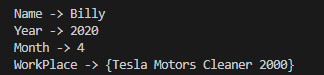


Рис.7. Результат PrintWorker

- *GetWorkersInfo()* – приймає множину типу *Worker* і повертає через вихідні параметри найбільшу та найменшу зарплату серед усіх працівників:

func GetWorkersInfo(arr []Worker) (int, int) {

    arrSalary := []int{}

*for* \_, value := *range* arr {

        arrSalary = append(arrSalary, value.WorkPlace.Salary)

    }

    max := arrSalary[0]

    min := arrSalary[0]

*for* \_, value := *range* arrSalary {

*if* value > max {

            max = value

        }

*if* value < min {

            min = value

        }

    }

*return* max, min

}

maxSalary, minSalary := GetWorkersInfo(workersArray)

    fmt.Println("maxSalary ->", maxSalary, "\nminSalary ->", minSalary)



Рис.8. Результат GetWorkersInfo

***Висновки:*** у ході виконання лабораторної роботи було засвоєно принципи проектування та оголошення структур та інтерфейсів; вивчено особливості реалізації структур та їх методів.