*осударственное образовательное учреждение высшего профессионального образования*



ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Системы обработки информации и управления» (ИУ-5)

**ЛАБОРАТОРНАЯ РАБОТА**

**№ 6**

\_\_\_\_\_\_\_\_\_\_\_\_\_Разработка на языке программирования Rust\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа ИУ5-35Б

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **16.12.2024** /**А. А. Торопыгин/** 

(Подпись, дата) (И.О.Фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /**Ю. Е. Гапанюк/**

(Подпись, дата) (И.О.Фамилия)

2024

### **Задание:**

1. Реализовать задачу 7 из ЛР №3-4 на языке Rust:

* В файле [data\_light.json](https://github.com/ugapanyuk/BKIT_2021/blob/main/notebooks/fp/files/data_light.json) содержится фрагмент списка вакансий.
* Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
* Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей.
* Функция f1 должна вернуть отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр.
* Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”.
* Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python”.
* Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

1. Разработать хотя бы один макрос.
2. Разработать модульные тесты (не менее 3 тестов).

**Текст программы:**

extern crate rand;

use serde\_json::{self, Value};

use std::{fs, ptr::null};

use rand::Rng;

macro\_rules! print\_vec {

($($arg:tt)\*) => {

for i in $($arg)\*.into\_iter(){ println!("{}", i); }

};

() => {

println("It's not a vector!");

}

}

fn f1(data: Value) -> Vec<String>{

let mut vec: Vec<String> = Vec::new();

let mut val: String;

for i in 0..data.as\_array().unwrap().len() {

if data[i]["job-name"].is\_null() {continue;}

val = data[i]["job-name"].to\_string().to\_lowercase();

val.retain(|c| c != '\"');

vec.push(val);

}

vec.sort();

vec.dedup();

return vec;

}

fn f2(vec: Vec<String>) -> Vec<String> {

let prog\_jobs = vec

.into\_iter()

.filter(|job| job.clone().starts\_with("программист"))

.collect::<Vec<String>>();

return prog\_jobs;

}

fn f3(vec: Vec<String>) -> Vec<String> {

let python\_exp = vec

.into\_iter()

.map(|job| format!("{} с опытом Python", job))

.collect::<Vec<String>>();

return python\_exp;

}

fn f4(vec: Vec<String>) -> Vec<String> {

let mut salary = rand::thread\_rng();

let rand\_salaries = vec

.into\_iter()

.map(|job| format!("{}, зарплата {} руб.",

job, salary.gen\_range(100000..200000)))

.collect::<Vec<String>>();

return rand\_salaries;

}

fn main() {

let str\_data = fs::read\_to\_string("data\_light.json")

.unwrap\_or\_else(|\_error| {panic!("Can't read file")});

let data: Value = serde\_json::from\_str(&str\_data).expect("Can't parse json");

print\_vec!(f4(f3(f2(f1(data)))));

}

#[cfg(test)]

mod tests {

use serde\_json::json;

use super::\*;

#[test]

fn test\_f1() {

let t = json!([

{"salary": 100, "job-name": "Я"},

{"salary": 100, "job-name": "А"},

{"job-name": "п"},

{"salary": 100}

]);

assert\_eq!(f1(t), vec![

"а".to\_string(),

"п".to\_string(),

"я".to\_string()

]);

}

#[test]

fn test\_f2() {

assert\_eq!(f2(vec!["программист".to\_string(), "не программист".to\_string()]),

vec!["программист".to\_string()]);

}

#[test]

fn test\_f3() {

assert\_eq!(f3(vec!["qwerty".to\_string(), "".to\_string()]),

vec!["qwerty с опытом Python".to\_string(), " с опытом Python".to\_string()])

}

}

**Вывод:**



