|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет»  Институт компьютерных наук и  технологий    **ОТЧЁТ**  по индивидуальной работе №1  по дисциплине «Языки программирования»  Вариант 5    Работу выполнил  студент группы ПМИ-10,2023-2024 1 курса  Казаков Н.С Фамилия И.О.  «18» января 2024 г.  Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |

**СОДЕРЖАНИЕ**

Постановказадачи...........................................................................................................................................3

Алгоритм решения..........................................................................................................................................4

Тестирование…................................................................................................................................................5

Код программы.............................................................................................................................................10

|  |  |
| --- | --- |
|  | |
|  | |
|  |  |
|  | |

**Постановка задачи**  
 Создать файл записей, в котором хранится информация о соискателях в кадровом агентстве:

фамилия, имя, отчество, пол, дата рождения (год, месяц, число), специальность, стаж работы

по специальности, знание иностранных языков, ожидаемый оклад. Разработать и реализовать

программу "Кадровое агентство", которая считывает исходную информацию и позволяет на основе

неё создавать следующие отчёты:

1. Полный список всех соискателей, который будет отсортирован следующему ключу: специальность

(по возрастанию) + фамилия (по возрастанию).

2. Список всех соискателей заданной специальности (вводится с клавиатуры), отсортированный

по следующему ключу: стаж работы (по убыванию) + пол (по убыванию) + фамилия (по возрастанию).

3. Список всех соискателей, претендующих на оклад в диапазоне от N1 до N2 рублей (вводится

с клавиатуры), отсортированный по следующему ключу: ожидаемый оклад (по убыванию) + фамилия

(по возрастанию).

Создать базу соискателей, включающую не менее 25 записей и на основе неё сформировать все

указанные списки (можно использовать вложенные списки и словари) и на основе неё сформировать

все указанные списки). Взаимодействие с базой данных обеспечивается отдельными функциями – добавление/удаление/изменение записи, вывода на экран всей базы и отдельно сформированных списков. Интерфейс пользователя удобнее организовать в виде меню с выбором соответствующего действия.

В программе обязательно наличие **дружественного интерфейса** и **защиты от некорректного пользовательского ввода.** База должна содержать такие записи, чтобы во всех списках явно прослеживался заданный вид сортировки по всем ключам. Для сортировки записей использовать Шейкерную сортировку.

**Алгоритм решения**

В предоставленном коде считываются данные из файла с именем "applicant.txt" и создает список словарей под названием "applicant" с информацией о каждом заявителе. Каждый словарь представляет кандидата и содержит следующие ключи: "фамилия", "имя", "отчество", "пол", "дата\_рождения", "специальность", "стаж", "языки" и "оклад". Он также сохраняет уникальные специальности в списке под названием "specialties ".

В коде определены три функции:

Функция "full\_list\_sorted"(applicants): возвращает отсортированный список всех applicants(претенденты), отсортированных по специальности (в порядке возрастания) и фамилии (в порядке возрастания).

Функция "filtered\_list\_by\_specialty (applicants, specialty)»: возвращает отсортированный список претендентов определенной специальности. Он принимает список кандидатов и специальность в качестве входных данных и сортирует кандидатов по опыту (в порядке убывания), полу (в порядке убывания) и фамилии (в порядке возрастания).

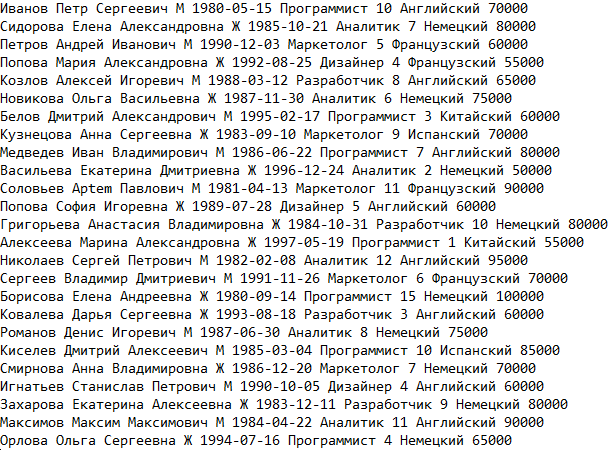
Функция "filtered\_list\_by\_salary\_range (applicants, min\_salary, max\_salary)": возвращает отсортированный список претендентов с зарплатой в пределах определенного диапазона. Он принимает список заявителей, минимальную и максимальную зарплату в качестве входных данных и сортирует заявителей по зарплате (в порядке убывания) и фамилии (в порядке возрастания).

Также программа предлагает пользователю ввести специальность из списка уникальных специальностей и создает файл "sorting\_by\_specialty.txt", содержащий и отсортированный список претендентов с выбранной специальностью. Он также предлагает пользователю ввести диапазон зарплат и создает файл "sorted\_by\_salary.txt", содержащий и отсортированный список претендентов в пределах этого диапазона зарплат.

**Тестирование:**

**1 Тест:**

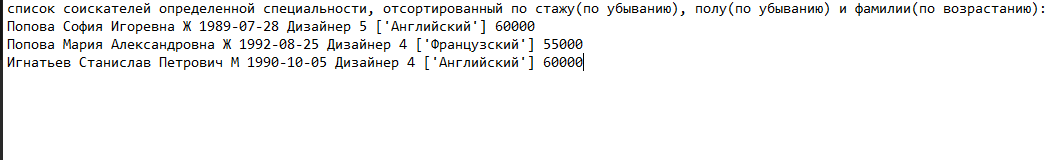
Входные данные:



Полный список всех соискателей, отсортированный по специальности (по возрастанию) + фамилии(по возрастанию)

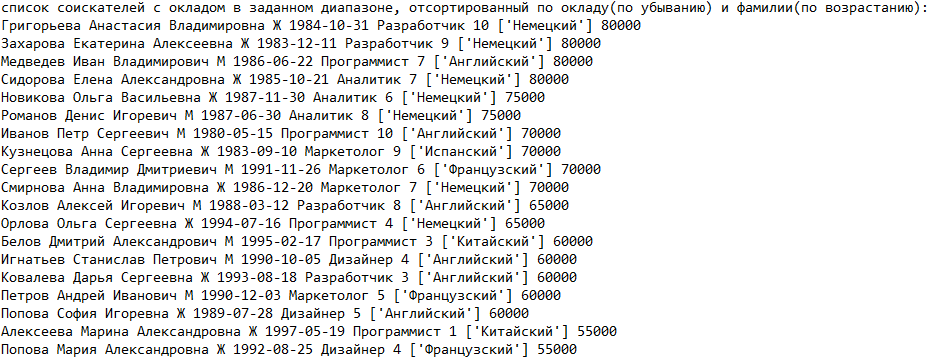


Список соискателей определенной специальности, отсортированный по стажу (по убыванию), полу(по убыванию) и фамилии(по возрастанию):

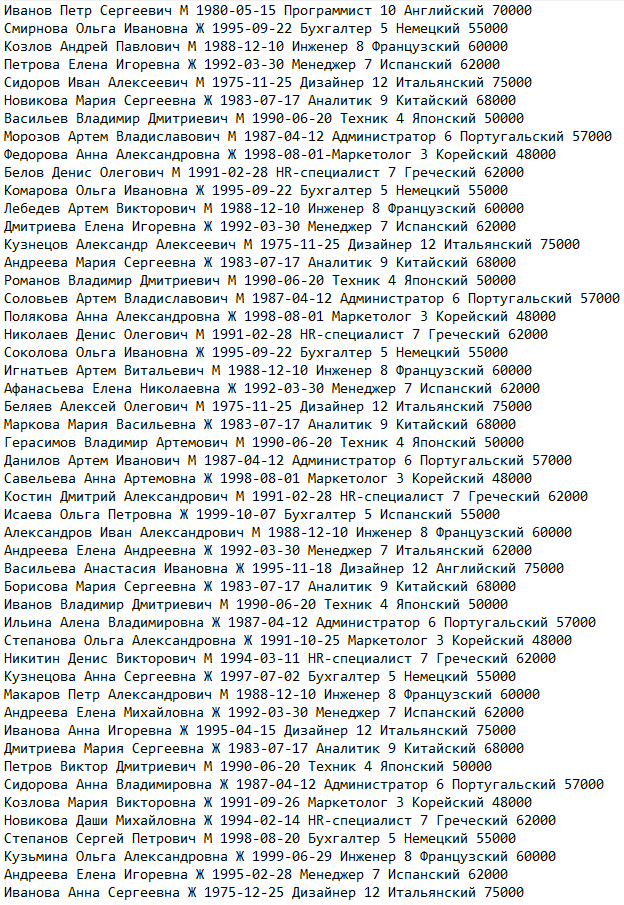


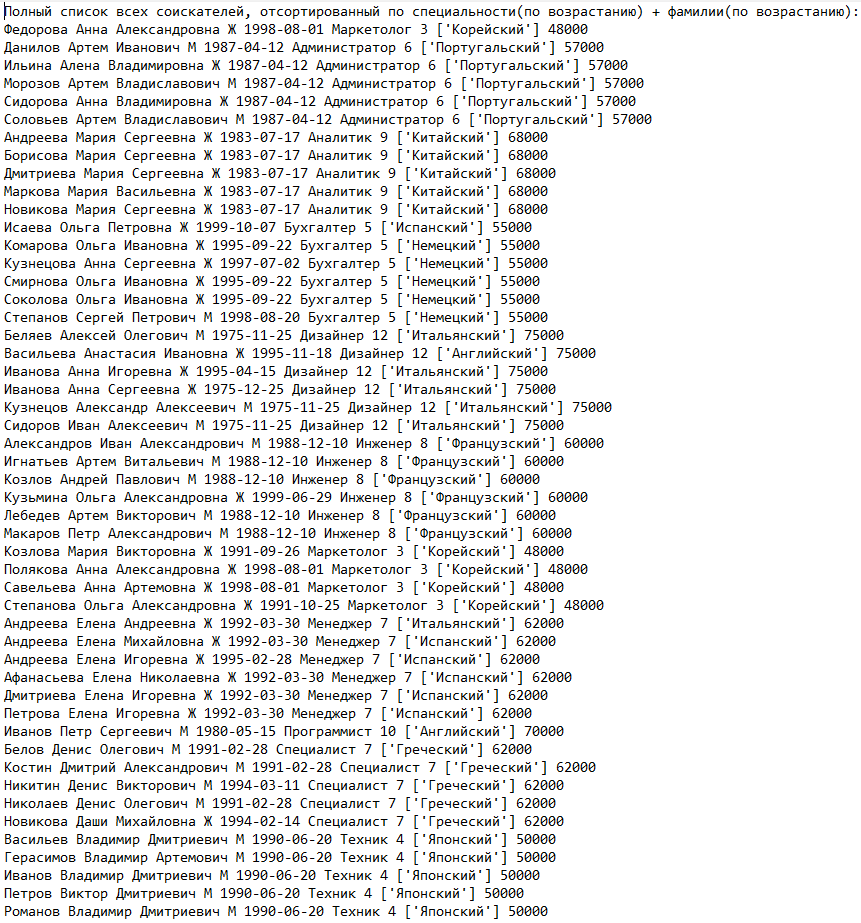
список соискателей с окладом в заданном диапазоне, отсортированный по окладу (по убыванию) и фамилии(по возрастанию):





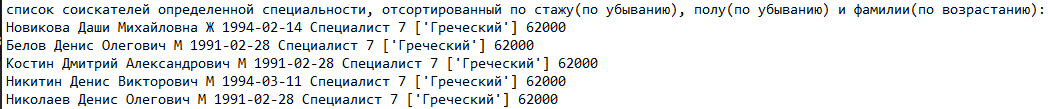
**2 Тест:**  
**Входные данные:**



Полный список всех соискателей, отсортированный по специальности (по возрастанию) + фамилии(по возрастанию)  


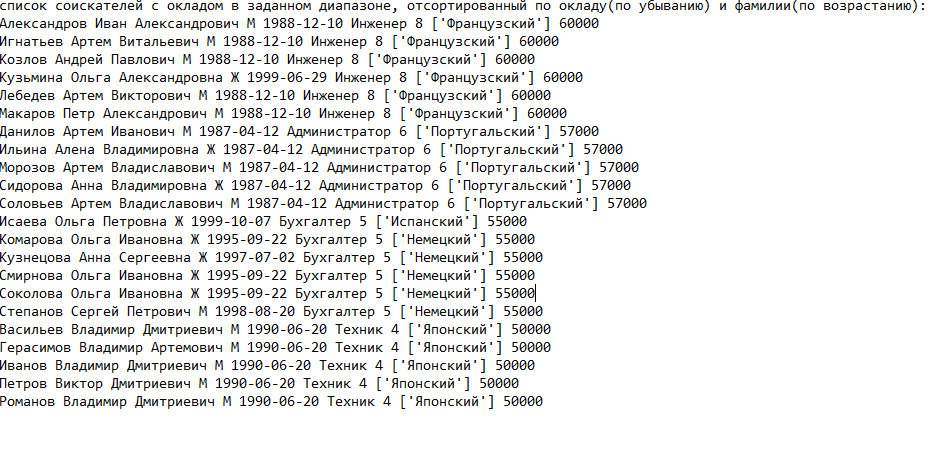
Список соискателей определенной специальности, отсортированный по стажу (по убыванию), полу(по убыванию) и

фамилии(по возрастанию):



Список соискателей с окладом в заданном диапазоне, отсортированный по окладу (по убыванию) и фамилии(по возрастанию):





**Код программы:**

applicants = []

specialties = []

with open('applicant.txt', 'r', encoding='utf-8') as file:

for line in file:

data = line.split() # Разделяем строку на части используя пробел в качестве разделителя

record = {

'фамилия': data[0],

'имя': data[1],

'отчество': data[2],

'пол': data[3],

'дата\_рождения': data[4],

'специальность': data[5],

'стаж': int(data[6]), # Преобразуем строку со стажем в целое число

'языки': data[7].split(','), # Преобразуем строку с языками в список

'оклад': int(data[8]) # Преобразуем строку с окладом в целое число

}

if data[5] not in specialties:

specialties.append(data[5])

applicants.append(record)

# Функция для создания отчета: полный список всех соискателей, отсортированный по специальности + фамилии

def full\_list\_sorted(applicants):

sorted\_applicants = sorted(applicants, key=lambda x: (x['специальность'], x['фамилия']))

return sorted\_applicants

# Функция для создания отчета: список соискателей определенной специальности, отсортированный по стажу, полу и фамилии

def filtered\_list\_by\_specialty(applicants, specialty):

filtered\_applicants = [applicant for applicant in applicants if applicant['специальность'] == specialty]

sorted\_filtered = sorted(filtered\_applicants, key=lambda x: (-x['стаж'], x['пол'], x['фамилия']))

return sorted\_filtered

# Функция для создания отчета: список соискателей с окладом в заданном диапазоне, отсортированный по окладу и фамилии

def filtered\_list\_by\_salary\_range(applicants, min\_salary, max\_salary):

filtered\_applicants = [applicant for applicant in applicants if min\_salary <= applicant['оклад'] <= max\_salary]

sorted\_filtered = sorted(filtered\_applicants, key=lambda x: (-x['оклад'], x['фамилия']))

return sorted\_filtered

# Записываем отсортированные данные в файлы

with open('sorting\_by\_last\_name+specialty.txt', 'w', encoding='utf-8') as f\_out:

full\_sorted = full\_list\_sorted(applicants)

f\_out.write("Полный список всех соискателей, отсортированный по специальности(по возрастанию) + фамилии(по возрастанию):" + '\n')

for applicant in full\_sorted:

f\_out.write(' '.join(str(applicant[key]) for key in applicant) + '\n')

specialty = (input('введите специальность из списка ' +str(specialties) + ':'))

specialty = specialty.title().strip() #убераем все пробелы в слове и первую букву делаем заглавной

# открываем файл в который будем записывать результат отсортированного списка по определенной специальности

with open('sorting\_by\_specialty.txt', 'w', encoding='utf-8') as f\_out:

while (specialty not in specialties):

print('Данной специальности нет в списке,пожалуйста,введите специальность из списка ' + str(specialties) + ':')

specialty = input('введите специальность из списка: ')

specialty = specialty.title().strip()

sorted\_by\_specialty = filtered\_list\_by\_specialty(applicants, specialty)

f\_out.write("список соискателей определенной специальности, отсортированный по стажу(по убыванию), полу(по убыванию) и фамилии(по возрастанию):" + '\n')

for applicant in sorted\_by\_specialty:

f\_out.write(' '.join(str(applicant[key]) for key in applicant) + '\n')

while True: # Запускаем цикл пока мы не введем нормальный диапазон чисел

try:

min\_salary, max\_salary = map(int, input('введите диопазон желаемого оклада через пробел: ').split()) # Считываем введенное число с клавиатуры

if min\_salary>max\_salary:

print("Вы ввели диапазон желаемого оклада от",max\_salary,'до',min\_salary )

else:

print("Вы ввели диапазон желаемого оклада от",min\_salary,'до',max\_salary )

break

except ValueError:

print("Ошибка: введенное значение не является числом")

# открываем файл в который будем записывать результат отсортированного списка по определенному диапазону оклада

with open('sorted\_by\_salary.txt', 'w', encoding='utf-8') as f\_out:

if min\_salary>max\_salary:

sorted\_by\_salary = filtered\_list\_by\_salary\_range(applicants, max\_salary, min\_salary)

else:

sorted\_by\_salary = filtered\_list\_by\_salary\_range(applicants, min\_salary, max\_salary)

f\_out.write("список соискателей с окладом в заданном диапазоне, отсортированный по окладу(по убыванию) и фамилии(по возрастанию):" + '\n')

for applicant in sorted\_by\_salary:

f\_out.write(' '.join(str(applicant[key]) for key in applicant) + '\n')