|  |  |  |
| --- | --- | --- |
| 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 8 | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 1 курса  Нуждин А. В.  «25» Января 2023 г. |
| Работу проверил  Рубцова М. Б.  «25» Января 2023 г. |
| Пермь 2023 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 6](#_Toc153130029)

[Код программы 7](#_Toc153130030)

# Постановка задачи

Создать файл записей, в котором хранится информация о вакансиях в кадровом агентстве: должность, необходимый стаж работы, пол, образование, минимальный возраст, максимальный возраст, знание иностранных языков, минимальный оклад, наличие соцпакета, испытательный срок. Разработать и реализовать программу "Кадровое агентство", которая считывает исходную информацию и позволяет на основе неё создавать следующие отчёты:

1.      Полный список всех вакансий, который будет отсортирован следующему ключу: образование (по возрастанию) + должность (по возрастанию).

2.      Список всех вакансий с определенным испытательным сроком (например, не менее 2 месяцев (вводится с клавиатуры)), отсортированный по следующему ключу: испытательный срок (по убыванию) + необходимый стаж работы (по убыванию) + максимальный возраст (по возрастанию).

3.      Список всех вакансий, с окладом в диапазоне от N1 до N2 рублей (вводится с клавиатуры), отсортированный по следующему ключу: наличие соцпакета (по возрастанию) + испытательный срок (по убыванию).

Создать базу вакансий, 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 сортировку Хоара.

# Алгоритм решения

Открываем файл «System\_file.txt» на котором храниться база данных.

Создаём двумерный список и записываем в него базу данных.

Выводим приветствие и просим пользователя выбрать одну из функций программы:

1. Сортировка 1 (сортировка по образованию, повторяющиеся кадры отсортировать по должности)
2. Сортировка 2 (сортировка списка вакансий с определённый м испытательным сроком. Сортировка производится по испытательному сроку. Повторяющиеся кадры отсортировать по стажу работы. Если кадры повторяются ещё раз, отсортировать их по максимальному возрасту)
3. Сортировка 3 (сортировка списка вакансий с определённым окладом. Отсортировать по наличию социального пакета и, если кадры совпадают, отсортировать по испытательному сроку)
4. Добавить элемент в конец списка
5. Изменить определённый элемент в списке
6. Удалить определённый элемент в списке
7. Вывести весь список
8. Создание и управление пользовательских списков

**Сортировка 1:**

Функция принимает изначальный список вакансий.

Создаём словарь списков, ключами которого является образование кандидата.

Запоминаем в отдельные списки ключи словаря и должности.

Сортируем списки с ключами и должностями с помощью сортировки Хоара.

Заполняем новый список: проходимся по всем спискам и словарю и если в определённый момент обнаруживается совпадение, записываем кадр в новый список.

Выводим отсортированный список.

**Сортировка 2**:

Функция принимает исходный список и минимальное значение испытательного срока.

Сокращаем список до подходящих нам по испытательному сроку значений

Сортируем список по следующему шаблону:

* Создаём словарь списков и берём в качестве ключа испытательные сроки.
* Сортируем ключи с помощью сортировки Хоара
* Собираем новый список из отсортированных по ключу словарей.
* Возвращаем искомый список.

Создаём словарь списков с ключами по значениям социального пакета.

Сортируем по вышеуказанному алгоритму содержимое словаря по ключам значения опыта работы.

Создаём ещё один список с тем же ключом и вкладываем в него отсортированный ранее список.

Сортируем содержимое вложенных словарей по вышеуказанному шаблону. За ключ на этот раз берём значения максимального возраста.

Собираем из получившегося словаря итоговый список.

Выводим отсортированный список.

**Сортировка 3:**

Функция принимает исходный список, минимальное значение оклада и максимальное значение оклада.

Сокращаем список до подходящих нам значений оклада.

Создаём словарь списков, в качестве ключа берём значение социальных пакетов.

Сортируем список ключей, с помощью сортировки Хоара.

Переворачиваем список (требуется по условию задачи).

Сортируем содержимое словаря с [помощью](https://ru.wiktionary.org/wiki/%D1%81%20%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E) алгоритма из (Сортировка 2). В качестве ключа берём значение испытательного срока

Собираем итоговый список из отсортированного словаря

Выводим отсортированный список.

**Добавление элемента в конце списка:**

Просим пользователя ввести данные о новой вакансии.

Увеличиваем список на одну ячейку.

Записываем в этот список полученную от пользователя информацию.

**Изменение определённого элемента в списке:**

Просим пользователя указать позицию изменяемого элемента

Просим пользователя ввести данные о новой вакансии

Заменяем данные изменяемого элемента на данные, полученные от пользователя

**Удаление определённого элемента в списке:**

Просим пользователя указать позицию удаляемого элемента.

Удаляем элемент из списка командой list.pop(value)

**Вывести весь список:**

Вывести весь список.

**Создание и управление пользовательскими списками:**

Проверяем, существуют ли уже пользовательские списки.

Если нет, то просим пользователя указать номера вакансий, которые он хочет добавить в свой список. Если пользователь введёт 0, запись вакансий окончится.

Увеличить переменную, отвечающую за количество списков на один.

Если списки уже созданы, спросить пользователя, хочет ли он отобразить уже существующие списки или создать новый.

Если пользователь выбирает первый вариант — вывести список.

В противном случае возвращаемся к пункту два.

**Типы данных:**

В данной работе я использовал вложенный список для хранения базы данных. Причиной такого выбора стала большая гибкость данной структуры в сравнении с массивами numpy и словарями. Такой подход позволяет редактировать список на ходу, как динамическую переменную.

Часто использовался простой список, для временного хранения информации. Это позволило сортировать основной список, просто записав вложенные списки в правильном порядке.

Использовались словари и вложенные словари. Чаще всего они требовались, когда нужно было отсортировать определённый участок списка, не изменяя остальную часть. После этого словари раскрывались, соединяя разные отсортированные списки в один.

# Тестирование

# Сортировка 1 работает в штатном режиме без входных данных и всегда выдаёт ожидаемый результат: правильно отсортированный список.

# Сортировка 2

|  |  |  |
| --- | --- | --- |
| **Введено** | **Ожидалось (По должностям)** | **Результат(По должностям)** |
| 3 д | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург, Каменщик, Генералиссимус, Комиссар, Святой Отец, Клерк, Клерк, Почтальон. | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург, Каменщик, Генералиссимус, Комиссар, Святой Отец, Клерк, Клерк, Почтальон. |
| 7 н | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург, Каменщик, Генералиссимус, Комиссар | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург, Каменщик, Генералиссимус, Комиссар |
| 7 м | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург | Император, Святой, Отец, Мать, Ветеринар, Адмирал, Святая Мать, Маршал, Канонисса, Металлург |
| 17 м | Император, Святой, Отец, Мать, Ветеринар, Адмирал | Император, Святой, Отец, Мать, Ветеринар, Адмирал |
| 11 г | Император, Святой, Отец | Император, Святой, Отец |

# Вывод: из представленных выше результатов можно сделать вывод, что сортировка работает в штатном режиме и стабильно выдаёт правильный ответ.

# Сортировка 3

|  |  |  |
| --- | --- | --- |
| **Введено** | **Ожидалось (По должностям)** | **Результат (По должностям)** |
| 1  100 | Отец(1), Мать(1), Фитс(100), Доктор (10)  Клерк(10), Святой(1) | Отец, Мать, Фитс, Доктор, Клерк, Святой |
| 100  1000 | Рыцарь, Фитс, Адмирал, Канонисса, Металлург, Святой отец, Оператор, Святая мать, Каменщик. | Рыцарь, Фитс, Адмирал, Канонисса, Металлург, Святой отец, Оператор, Святая мать, Каменщик. |
| 1000  10000 | Рыцарь, Клерк(1000), Адмирал, Канонисса, Металлург, Святой отец, Почтальон, Оператор, Святая мать, Каменщик, Крысолов | Рыцарь, Клерк, Адмирал, Канонисса, Металлург, Святой отец, Почтальон, Оператор, Святая мать, Каменщик, Крысолов |
| 10000  100000 | Комиссар, Клерк, Почтальон, Клерк, Крысолов, Генералиссимус. | Комиссар, Клерк, Почтальон, Клерк, Крысолов, Генералиссимус. |
| 100000  1000000 | Комиссар, Ветеринар, Клерк, Маршал, Генералиссимус. | Комиссар, Ветеринар, Клерк, Маршал, Генералиссимус. |

Вывод: Сортировка 3 выдаёт стабильно правильный ответ в зависимости от входных данных

Работа с пользовательскими списками:

|  |  |  |
| --- | --- | --- |
| Введено | Ожидание | Результат |
| 1,3,4,7,0 | Врач, Клерк, Клерк, Каменщик | Врач, Клерк, Клерк, Каменщик |
| Получить пользовательский список 1. | Врач, Клерк, Клерк, Каменщик | Врач, Клерк, Клерк, Каменщик |
| 4, 16, 3, 0 | Клерк, Рыцарь, Клерк | Клерк, Рыцарь, Клерк. |
| 225, 3 | Вы ввели некорректные данные. Такой вакансии нет. | Вы ввели некорректные данные. Такой вакансии нет. |

Тестирование окончено. Все три сортировки выдают стабильно правильный результат.

# **Код программы**

import random

def HoarSort\_symbols(A):#Сортировка строк по алфавиту

if len(A) <= 1:

return A

else:

q = random.choice(A)

L = []

M = []

R = []

for elem in A:

if elem < q:

L.append(elem)

elif elem > q:

R.append(elem)

else:

M.append(elem)

return HoarSort\_symbols(L) + M + HoarSort\_symbols(R)

def HoarSort\_numbers(A):#Сортировка массива от минимума до максимума

B=[]

C=[]

for i in A:

B.append(int(i))

if len(B) <= 1:

return B

else:

q = random.choice(B)

L = []

M = []

R = []

for elem in B:

if elem < q:

L.append(elem)

elif elem > q:

R.append(elem)

else:

M.append(elem)

return HoarSort\_numbers(L) + M + HoarSort\_numbers(R)

def time(time):#Приведение чисел к одному значению

temp=time.split(" ")

try:

if temp[1]=="д":

return int(temp[0])

elif temp[1]=="н":

return int(temp[0])\*7

if temp[1]=="м":

return int(temp[0])\*30

elif temp[1]=="г":

return int(temp[0])\*30\*12

return 0

except:

return 0

def prin(lists, side=1):#Вывести

print("\nДолжность|необходимый стаж работы|пол|образование|минимальный возраст|максимальный возраст|знание иностранных языков|минимальный оклад|наличие соцпакета|испытательный срок\n")

if side == 1:#По возрастанию

for i in lists:

for j in i:

print(j,end=" | ")

print("\n")

else:#По убыванию

for i in range(len(lists)-1,0,-1):

for j in range(len(lists[i])):

print(lists[i][j],end=" | ")

print("\n")

def sorted\_education\_post(rec):#Сортировка по образованию + должности

temp\_list=[]

temp\_list1=[]

temp\_dict=dicts(rec,3)#Создаём словарь профессий по ключу образования

for i in temp\_dict.keys():#Создаём список должностей и образования

for j in temp\_dict[i]:

temp\_list.append(j[0])

for i in rec:

temp\_list1.append(i[3])

temp\_list=HoarSort\_symbols(temp\_list)#Сортируем заранее созданные списки

temp\_list1=HoarSort\_symbols(temp\_list1)

lists=[]

for i in temp\_list1:#создаём новый лист на основе отсортированых списков

for j in temp\_list:

for c in temp\_dict[i]:

if c[0]==j and not(c in lists):

lists.append(c)

break

prin(lists)#Выводим

def sorted\_period(A,N,mode="time",side="left"):#Сортировка списка профессий по числам

if len(A)>1:

temp\_list=[]

temp\_dict=dict()

if mode=="time":#Создаём словарь с ключом в виде точного времени(д н м г)

for i in A:

if time(i[N]) in temp\_dict.keys():

temp\_dict[time(i[N])].append(i[0])

else:

temp\_dict[time(i[N])]=[i[0]]

temp\_list.append(time(i[N]))

else:#Создаём словарь с ключом в виде неточного времени

for i in A:

temp=i[N].split(" ")

if temp[0]in temp\_dict.keys():

temp\_dict[temp[0]].append(i[0])

else:

temp\_dict[temp[0]]=[i[0]]

temp\_list.append(temp[0])

lists=[]

temp\_list=HoarSort\_numbers(temp\_list)#Сортируем ключи

for i in temp\_list:#Собираем отсортированый список

for j in A:

try:

for c in range(len(temp\_dict[i])):

if j[0]==temp\_dict[i][c] and (j in lists)==False:

lists.append(j)

break

except:

for c in range(len(temp\_dict[str(i)])):

if j[0]==temp\_dict[str(i)][c] and (j in lists)==False:

lists.append(j)

break

if side=="left":#возвращает список по возрастанию

return lists

else:#возвращает список по убыванию

lists.reverse()

return lists

else:

return A

def sorted\_salary(lists,N1,N2):#Сортировка по соцпакету и исп. сроку

temp\_list=[]

temp\_list1=[]

for i in range(1,len(lists)):#Отсеиваем нужный список и запоминаем встреченые пакеты

temp=lists[i][7].split(" ")

try:

t=int(temp[0])

except:

t=N1-1

if t>=N1 and t<=N2:

temp\_list.append(lists[i])

if not(lists[i][8] in temp\_list1):

temp\_list1.append(lists[i][8])

temp\_dict=dicts(temp\_list,8)#Создаём словарь по ключу соц пакетов

temp\_list1=HoarSort\_symbols(temp\_list1)#Сортируем ключи и преворачиваем список

temp\_list1.reverse()

temp\_list=[]

for i in temp\_list1:#собираем, попутно проводя сортировку по испытательному сроку

temp\_list2=[]

for j in temp\_dict[i]:

temp\_list2.append(j)

temp\_list+=sorted\_period(temp\_list2,9,"time","right")

prin(temp\_list)

def dicts(temp\_list,N):#Создание словоря с ключом N

temp\_dict=dict()

for i in temp\_list:

if i[N] in temp\_dict.keys():

temp\_dict[i[N]].append(i)

else:

temp\_dict[i[N]]=[i]

return temp\_dict

def sorted\_period\_experience\_age(lists,N):#Сортировка по исп. сроку + опыту + возрасту

temp\_list=[]

temp\_list1=[]

temp\_dict=dict()

temp\_dict2=dict()

for i in range(1,len(lists)):#Изымаем из списка подходящие элементы

temp=time(lists[i][9])

if temp>time(N):

temp\_list.append(lists[i])

temp\_list=sorted\_period(temp\_list,9)#Сортеруем список по испытательному сроку

temp\_dict=dicts(temp\_list,9)#Создаём словарь с ключами по испытательному сроку

for i in temp\_dict.keys():#Сортеруем по опыту работы и помещаем в новый список

temp\_dict[i]=sorted\_period(temp\_dict[i],1," ")

temp\_dict2[i]=dicts(temp\_dict[i],1)

temp\_list=[]

for i in temp\_dict2.keys():#Собираем итоговый список и сортеруем его

for j in temp\_dict2[i].keys():

temp\_list+=sorted\_period(temp\_dict2[i][j],5," ","right")

prin(temp\_list," ")

def dda(N):#продолжение добавления элемента

while(True):

try:

temp=input("Введите "+N+": ")

return temp

except:

print("Что-то случилось на нашей стороне. Попробуйте ещё раз.")

def checking(temps):#Проверка соответствует ли значение критериям

try:

temp=temps.split(" ")

temp\_i=int(temp[0])

return True

except:

print("Вы ввели некорректные данные. Вот образец коррекных данных: |Число| |значение|")

return False

def add(lists, position=0):#Добавление или замена элемента

temp\_list=[]

temp=0

temp\_i=0

temp\_list.append(dda("должность"))

temp=dda("необходимый стаж работы")

while not(checking(temp)):

temp=dda("необходимый стаж работы")

temp\_list.append(temp)

temp\_list.append(dda("пол"))

temp=dda("минимальный возраст")

while not(checking(temp)):

temp=dda("минимальный возраст")

temp\_list.append(temp)

temp=dda("максимальный возраст")

while not(checking(temp)):

temp=dda("максимальный возраст")

temp\_list.append(temp)

temp\_list.append(dda("знание иностранных языков"))

temp=dda("минимальный оклад")

while not(checking(temp)):

temp=dda("минимальный оклад")

temp\_list.append(temp)

temp\_list.append(dda("наличие соцпакета(да - имеется, друое - нет)"))

temp=dda("испытательный срок")

while not(checking(temp)):

temp=dda("испытательный срок")

temp\_list.append(temp)

if position==0:

lists.append(temp\_list)

else:

lists[position]=temp\_list

def new\_mass(lists,temp):#Создание нового пользовательского списка

temp\_list=[]

try:

for i in temp:

if not(lists[i] in temp\_list):

temp\_list.append(lists[i])

return temp\_list

except:

print("В списке нет элеменат под числом"+i)

text=open("System\_file.txt","r")

recruitment=[]

temp=text.readline()

while temp:

if "\n" in temp:

temp=temp[:-1]

recruitment.append(temp.split("/"))

temp=text.readline()

text.close()

users\_list=[]

te=1

ty=0

print("Добро пожаловать в кадровое агентство. Что вы хотите сделать сегодня?\n")

print("\n\nВведите 1 — чтобы отсортировать по образовании + должности (В алфовитном порядке)\nВведите 2 — чтобы отсортировать по испытательному сроку + необходимому стажу работы + максимальному возрасту (в определнном диапозонеиспытательного срока)\nВведите 3 — чтобы отсортировать по наличию социального пакета + испытательному сроку (в определнном диапозоне оклада)\nВведите 4 - чтобы добавить новый элемент\nВведите 5 - чтобы изменить элемент таблици\nВведите 6 - чтобы удалить элемент таблици\nВведите 7 - чтобы вывести таблицу\nВведите 8 - чтобы работать с пользовательскими списками\nВведите 0 - чтобы выйти")

try:

temp=input("Введите новую команду: ")

except:

temp="7"

while temp!="0":

if temp=="1":

sorted\_education\_post(recruitment)#Сортировка по образованию и должности

elif temp=="2":#Сортировка по исп. сроку и опыту и М. возрасту

try:

year=input("Введите минимальный испытательный срок(сначала число, затем, через пробел, условное обозначение: д — день, н — неделя, м — месяц, г — год): ")

sorted\_period\_experience\_age(recruitment,year)

except:

print("Вы ввели некоректные данные!")

elif temp=="3":#Сортировка по соцпакету и исп. сроку

try:

N1=int(input("Введите минимальный оклад: "))

N2=int(input("Введите максимальный оклад: "))

sorted\_salary(recruitment,N1,N2)

except:

print("Вы ввели некоректные данные!")

elif temp=="4":#Добавление элемента

add(recruitment)

elif temp=="5":#Замена элемента

try:

pos=int(input("Введите позицию изменяемого элемента (начаная с единици): "))

add(recruitment,pos-1)

except:

print("Вы ввели некоректные данные!")

elif temp =="6":#Удаление элемента

try:

pos=int(input("Введите позицию удаляемого элемента (начаная с единици): "))

recruitment.pop(pos-1)

except:

print("Вы ввели некоректные данные!")

elif temp=="7":#Выведение всего списка

temp\_list=[]

print("Изначальная таблица\n")

prin(recruitment)

elif temp=="8":#Работа с пользовательскими списками

try:

if not(users\_list):#Создание пользовательского списка

te=int(input("Введите номер нового элемента пользовательского списка(начаная с одного) для завершения введите 0: "))

while(te!=0):

temp\_list.append(te)

te=int(input("Введите номер нового элемента пользовательского списка(начаная с одного) для завершения введите 0: "))

users\_list.append(new\_mass(recruitment,temp\_list))

prin(users\_list[0])

ty+=1

else:#Спросить пользователя, что делат: вывести существующий список или создать новый

te=int(input("Введите номер пользовательского списка от 1 до "+str(ty)+", чтобы вывести его. Или 0, чтобы создать новый: "))

if te!=0:

prin(users\_list[te-1])

else:

te=int(input("Введите номер нового элемента пользовательского списка(начаная с одного) для завершения введите 0: "))

while(te!=0):

temp\_list.append(te-1)

te=int(input("Введите номер нового элемента пользовательского списка(начаная с одного) для завершения введите 0: "))

users\_list.append(new\_mass(recruitment,temp\_list))

prin(users\_list[ty])

ty+=1

except:

print("Вы ввели некоректные данные.")

else:

try:

temp\_1=int(temp)#Если пользователь ошибся, сообщеть об этом

print("Команды "+temp+" нет. Возможно вы подразумивали "+ temp\_1)

except:

print("Команды "+temp+" нет.")

print("\n\nВведите 1 — чтобы отсортировать по образовании + должности (В алфовитном порядке)\nВведите 2 — чтобы отсортировать по испытательному сроку + необходимому стажу работы + максимальному возрасту (в определнном диапозонеиспытательного срока)\nВведите 3 — чтобы отсортировать по наличию социального пакета + испытательному сроку (в определнном диапозоне оклада)\nВведите 4 - чтобы добавить новый элемент\nВведите 5 - чтобы изменить элемент таблици\nВведите 6 - чтобы удалить элемент таблици\nВведите 7 - чтобы вывести таблицу\nВведите 8 - чтобы работать с пользовательскими списками\nВведите 0 - чтобы выйти")

try:

temp=input("Введите новую команду: ")

except:#На всякий случай

print("Что-то случилось на нашей стороне. Попробуйте ещё раз.")