Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №3**

Дисциплина: «Разработка профессиональных приложений»

Файлы и словари

Вариант №8

Выполнил:

студент группы ИВТАСбд-21

Зюзин Г.А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту.**

История посещений поликлиники: №, ФИО пациента, ФИО врача, причина обращения, длительность.

Считайте информацию из файла в соответствующую структуру (словарь):  
1. Выведите информацию об объектах, отсортировав их по одному полю (строковому);  
2. Выведите информацию об объектах, отсортировав их по одному полю (числовому);  
3. Выведите информацию, соответствующую какому-либо критерию;  
4. Добавьте к программе возможность сохранения новых данных обратно в файл.

**Описание реализации.**  
 В данной программе реализована функциональность для работы с историей посещений поликлиники. В начале программы определены функции и переменные, затем следуют рычаги для управления программой.

Основной функционал программы включает:

1. Чтение данных из CSV-файла в словарь:

* Создается пустой список mydicts, представляющий историю посещений.
* Открывается CSV-файл с помощью csv.DictReader.
* Каждая строка из файла считывается и добавляется в список mydicts в виде словаря.

1. Сортировка записей по одному из полей:

* Функция my\_sort принимает список mydicts и выбранный способ сортировки (way\_sort).
* В зависимости от выбранного способа сортировки, используя лямбда-функции, происходит сортировка списка mydicts по соответствующему полю.

1. Вывод информации о записях:

* Функция format\_print\_dict принимает словарь line и выводит информацию о записи, включая номер, ФИО пациента, ФИО врача, причину обращения и длительность.
* После чтения данных из файла, пользователю предлагается выбрать способ сортировки записей.
* Запускается функция my\_sort для сортировки записей по выбранному способу.
* Затем каждая отсортированная запись выводится с помощью функции format\_print\_dict.

1. Добавление новой записи и запись ее в файл:

* Если в рычагах управления программой установлен флаг WRITE\_RECORD в значение True, пользователю предлагается ввести данные новой записи.
* Затем создается словарь line с информацией о новой записи.
* Открывается CSV-файл в режиме добавления записей ("a") с помощью csv.DictWriter.
* С помощью объекта писателя writer происходит запись новой строки в файл.

**Описание возникших затруднений.**

В ходе выполнения лабораторной работы могут возникнуть следующие затруднения:

* При указании пути к директории для подсчета файлов (GET\_COUNT\_FILE\_DIR), необходимо убедиться, что указанный путь существует и содержит нужные файлы.
* При записи новой записи в файл (WRITE\_RECORD), необходимо внимательно заполнить данные и убедиться, что файл существует и имеет права на запись.
* При использовании способов сортировки (way\_sort), необходимо выбрать одно из доступных значений от 0 до 4.

**Описание альтернативных способов решения.**

Возможно, решение этой задачи можно реализовать при помощи массивов ключей и значений.

**Листинг кода.**

# Лабораторная работа №3  
# Вариант 8  
import os  
import csv  
import datetime  
  
mydicts = [] # Словарь  
path = "lab3.csv"  
  
# Сортировка различные варианты  
def my\_sort(mydicts, way\_sort):  
 if way\_sort == 0:  
 mydicts.sort(key=lambda x: int(x['id'])) # Сортировка по id  
 elif way\_sort == 1:  
 mydicts.sort(key=lambda x: x['name\_p']) # Сортировка по имени пациента  
 elif way\_sort == 2:  
 mydicts.sort(key=lambda x: x['name\_d']) # Сортировка по врача  
 elif way\_sort == 3:  
 mydicts.sort(key=lambda x: x['reason']) # Сортировка по причине обращения  
 elif way\_sort == 4:  
 mydicts.sort(key=lambda x: x['time']) # Сортировка по времени приёма (по возрастанию)  
 else:  
 print("Такого метода сортировки нет")  
 quit()  
  
# Функция для преобразовния секунд в формат минута:секунда  
def get\_format(seconds: int):  
 minutes = str(seconds // 60)  
 seconds = str(seconds % 60)  
  
 if int(minutes) < 10:  
 minutes = '0' + minutes  
 if int(seconds) < 10:  
 seconds = '0' + seconds  
  
 return f"{minutes}:{seconds}"  
  
# Функция для вывода записи  
def format\_print\_dict(line):  
 print(f"\nЗапись № {line['id']}:"  
 f"\nПациент - {line['name\_p']}"  
 f"\nВрач - {line['name\_d']}"  
 f"\nПричина обращения - {line['reason']}"  
 f"\nДлительность обращения - {line['time']}\n")  
  
  
# Рычаги для управления программы  
GET\_COUNT\_FILE\_DIR = False # Считать файлы в директории  
WRITE\_RECORD = False # Сделать запись в csv файл  
LOAD\_RECORDS = True # Загрузить записи  
OUTPUT\_RECORDS = True # Вывести записи, выбрав сбособ сортировки. LOAD\_RECORS должен быть True  
SELECT\_RECORDS = False # Вывести записи по одному из критериев. LOAD\_RECORS должен быть True  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 if GET\_COUNT\_FILE\_DIR:  
 try:  
 path = input("Введите путь директории(папки): ")  
 # root - имя корневого каталога  
 # dirs - список имен вложенных папок  
 # files - список файлов в текущем каталоге  
 root, dirs, files = next(os.walk(path)) # Выдаёт тройной кортеж - (dirpath, dirnames, filenames)  
 # возвращающает имена файлов в дереве каталогов, двигаясь по дереву сверху вниз  
 print("Количество файлов в данной директории: ", len(files))  
 except FileNotFoundError:  
 print("Директория с таким путём не найдена...")  
 quit()  
  
 if WRITE\_RECORD:  
 name\_patient = input("ФИО пациента: ")  
 name\_doc = input("ФИО врача: ")  
 reason = input("Причина обращения: ")  
 diraction = int(input("Длительность приёма(сек): "))  
 time = get\_format(diraction)  
 idx = datetime.datetime.now().strftime("%d%H%M%S")  
  
 line = {'id': idx, 'name\_p': name\_patient, 'name\_d': name\_doc, 'reason': reason, 'time': time}  
 print(f"Записи присвоен № {line['id']}.")  
  
 with open(path, "a", newline="") as file:  
 writer = csv.DictWriter(file, fieldnames=['id', 'name\_p', 'name\_d', 'reason', 'time']) # Объект писателя, с помощью которого и записываем файл  
 writer.writerow(line) # Сделать запись  
 file.close()  
  
 if LOAD\_RECORDS:  
 with open(path, "r") as tabel\_file:  
 reader = csv.DictReader(tabel\_file, fieldnames=['id', 'name\_p', 'name\_d', 'reason', 'time'])  
 for record in reader:  
 mydicts.append(dict(record))  
  
 if OUTPUT\_RECORDS:  
 print("Отсоритровать записпи по\n0 - id\n1 - ФИО пациента\n"  
 "2 - ФИО врача\n3 - причине обращения\n4 - длительности посещения")  
 way = int(input())  
 my\_sort(mydicts, way)  
  
 for line in mydicts:  
 format\_print\_dict(line)  
  
 if SELECT\_RECORDS:  
 for line in mydicts:  
 if(line['reason'] == "Temperature"):  
 format\_print\_dict(line)

**Вывод.**

В результате выполнения лабораторной работы была разработана программа, позволяющая работать с историей посещений поликлиники. Она позволяет считывать данные из CSV-файла в словарь, сортировать записи по различным полям, выводить информацию о записях и добавлять новые записи с последующей записью в файл.

Реализация программы включает использование функций для сортировки и вывода данных, а также чтения и записи в CSV-файл. Весь функционал программы был успешно реализован без возникновения затруднений.

Таким образом, цели работы были достигнуты, и программа готова для использования при работе с историей посещений поликлиники.