МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет по лабораторной работе №3**

**Вариант 12**

По дисциплине: «Скриптовые языки программирования»

**«Работа с файлами»**

Выполнил:

студент группы

10702423

Татур Е. Ю.

Проверил**:**

ст.преподаватель

Козловская Ю.Б.

Минск 2024

**Цель работы**

Выполнить подробным образом задания из разных уровней, согласно варианта.

Выполните задания своего варианта с подробными пояснениями:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | **Номера заданий** | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| 12. | 14 | 32 | 40 | 52 | 53 | 54e | 54b | 54f |

**Задания**

14. Создайте файл с марками автомобилей. Расположите каждое значение в строку через «-».

32. Создайте файл с перечнем мебели. Расположите каждое значение на отдельной строке.

40. Создайте файл с названиями фильмов. Расположите по 3 значения в отдельной строке.

52. Запишите в файл ваше любимое стихотворение. Выведите его на экран и укажите, каких слов в нем больше: начинающихся на гласную или на согласную (регистр букв не учитывать).

53. Создайте текстовый файл, в котором хранится информация о занятости мест в зрительном зале в двоичном виде (0 – свободно, 1 – занято). Количество рядов рассчитывается по формуле: возраст одного из родителей минус возраст студента. Количество мест в ряду – 15. Напишите программу, которая позволит пользователю увидеть количество свободных мест, а также, введя номер ряда и места, получить информацию, свободно или занято место.

54. Создайте телефонный справочник в формате JSON. Напишите программу, которая позволяет (предоставляет функции):

e. удалять контакты по имени;

b. выполнять поиск контактов по номеру телефона;

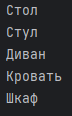
f. удалять номер телефона из контактов;

**Ход выполнения работы**

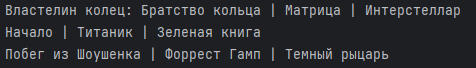
**Результат работы**



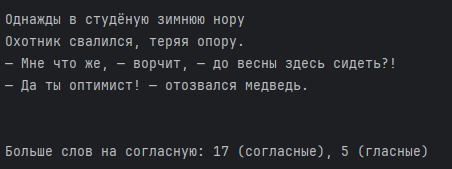
*Рисунок 1 – Результат выполненного задания 14.*



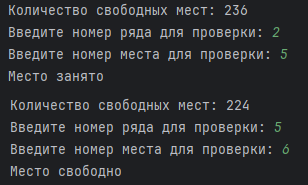
*Рисунок 2 – Результат выполненного задания 32.*



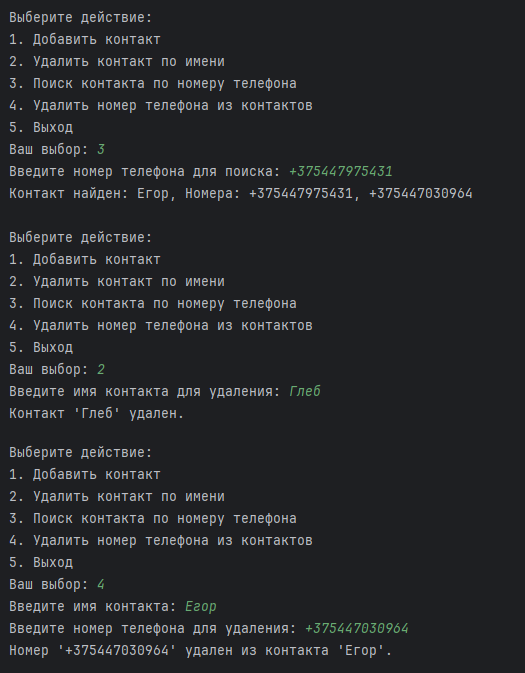
*Рисунок 3 – Результат выполненного задания 40.*



*Рисунок 4 – Результат выполненного задания 52.*



*Рисунок 5 – Результат выполненного задания 53.*



*Рисунок 6 – Результат выполненного задания 54.*

**Листинг**

**ex14.py:**

# Функция для записи марок автомобилей в файл  
def write\_car\_brands\_to\_file(filename, car\_brands):  
 with open(filename, 'w') as file:  
 file.write(' - '.join(car\_brands))  
  
  
# Функция для чтения содержимого файла и вывода на экран  
def read\_and\_print\_file(filename):  
 with open(filename, 'r') as file:  
 brand = file.read()  
 print(brand)  
  
  
# Укажите имя файла  
filename = 'car\_brands.txt'  
  
# Список марок автомобилей  
car\_brands = ['Toyota', 'Honda', 'Ford', 'BMW', 'Mercedes']  
  
# Запись марок автомобилей в файл  
write\_car\_brands\_to\_file(filename, car\_brands)  
  
# Чтение содержимого файла и вывод на экран  
read\_and\_print\_file(filename)

**ex32.py:**

# Функция для записи перечня мебели в файл  
def write\_furniture\_to\_file(filename, furniture):  
 with open(filename, 'w') as file:  
 for item in furniture:  
 file.write(item + '\n')  
  
  
# Функция для чтения содержимого файла и вывода на экран  
def read\_and\_print\_file(filename):  
 with open(filename, 'r') as file:  
 content = file.read()  
 print(content)  
  
  
# Укажите имя файла  
filename = 'furniture.txt'  
  
# Список мебели  
furniture = ['Стол', 'Стул', 'Диван', 'Кровать', 'Шкаф']  
  
# Запись перечня мебели в файл  
write\_furniture\_to\_file(filename, furniture)  
  
# Чтение содержимого файла и вывод на экран  
read\_and\_print\_file(filename)

**ex40.py:**

# Функция для записи названий фильмов в файл  
def write\_movies\_to\_file(filename, movies):  
 with open(filename, 'w') as file:  
 for i in range(0, len(movies), 3): # Проходим по списку с шагом 3  
 line = movies[i:i + 3] # Берем подсписок из трех элементов  
 file.write(' | '.join(line) + '\n') # Записываем их в файл, разделяя символом ' | '  
  
  
# Функция для чтения содержимого файла и вывода на экран  
def read\_and\_print\_file(filename):  
 with open(filename, 'r') as file:  
 movie = file.read()  
 print(movie)  
  
  
# Укажите имя файла  
filename = 'movies.txt'  
  
# Список названий фильмов  
movies = [  
 'Властелин колец: Братство кольца',  
 'Матрица',  
 'Интерстеллар',  
 'Начало',  
 'Титаник',  
 'Зеленая книга',  
 'Побег из Шоушенка',  
 'Форрест Гамп',  
 'Темный рыцарь'  
]  
  
# Запись названий фильмов в файл  
write\_movies\_to\_file(filename, movies)  
  
# Чтение содержимого файла и вывод на экран  
read\_and\_print\_file(filename)

**ex52.py:**

# Функция для записи стихотворения в файл  
def write\_poem\_to\_file(filename, poem):  
 with open(filename, 'w') as file:  
 file.write(poem)  
  
  
# Функция для чтения содержимого файла и вывода на экран  
def read\_and\_print\_file(filename):  
 with open(filename, 'r') as file:  
 poem = file.read()  
 print(poem)  
 return poem  
  
  
# Функция для подсчета слов по первой букве  
def count\_words\_by\_first\_letter(poem):  
 vowels = 'аеёиоуыэюяАЕЁИОУЫЭЮЯ' # Гласные буквы  
 consonants = 'бвгджзйклмнпрстфхцчшщБВГДЖЗЙКЛМНПРСТФХЦЧШЩ' # Согласные буквы  
  
 vowel\_count = 0  
 consonant\_count = 0  
  
 words = poem.split() # Разделяем стихотворение на слова  
  
 for word in words:  
 first\_letter = word[0] # Берем первую букву слова  
 if first\_letter in vowels:  
 vowel\_count += 1  
 elif first\_letter in consonants:  
 consonant\_count += 1  
  
 return vowel\_count, consonant\_count  
  
  
# Укажите имя файла  
filename = 'poem.txt'  
  
# Стихотворение (пример)  
poem = """   
Однажды в студёную зимнюю нору   
Охотник свалился, теряя опору.   
— Мне что же, — ворчит, — до весны здесь сидеть?!   
— Да ты оптимист! — отозвался медведь.  
"""  
  
# Запись стихотворения в файл  
write\_poem\_to\_file(filename, poem)  
  
# Чтение содержимого файла и вывод на экран  
read\_and\_print\_file(filename)  
  
# Подсчет слов по первой букве  
vowel\_count, consonant\_count = count\_words\_by\_first\_letter(poem)  
  
# Вывод результатов подсчета  
if vowel\_count > consonant\_count:  
 print(f"\nБольше слов на гласную: {vowel\_count} (гласные), {consonant\_count} (согласные)")  
elif consonant\_count > vowel\_count:  
 print(f"\nБольше слов на согласную: {consonant\_count} (согласные), {vowel\_count} (гласные)")  
else:  
 print(f"\nКоличество слов одинаковое: {vowel\_count} (гласные), {consonant\_count} (согласные)")

**ex53.py:**

import random  
  
  
# Функция для создания файла с занятостью мест  
def create\_seating\_chart(filename):  
 rows = 48 - 18 # Количество рядов  
 seats\_per\_row = 15  
 seating\_chart = []  
  
 for \_ in range(rows):  
 row = [random.choice([0, 1]) for \_ in range(seats\_per\_row)]  
 seating\_chart.append(row)  
  
 # Запись в файл  
 with open(filename, "w") as file:  
 for row in seating\_chart:  
 file.write("".join(map(str, row)) + "\n")  
  
  
# Функция для подсчета свободных мест  
def count\_free\_seats(filename):  
 with open(filename, "r") as file:  
 lines = file.readlines()  
  
 free\_count = sum(row.count('0') for row in lines)  
 return free\_count  
  
  
# Функция для проверки состояния конкретного места  
def check\_seat(filename, row\_number, seat\_number):  
 with open(filename, "r") as file:  
 lines = file.readlines()  
  
 if row\_number < 1 or row\_number > len(lines):  
 return "Неверный номер ряда"  
  
 row = lines[row\_number - 1].strip()  
 if seat\_number < 1 or seat\_number > len(row):  
 return "Неверный номер места"  
  
 if row[seat\_number - 1] == '0':  
 return "Место свободно"  
 else:  
 return "Место занято"  
  
  
# Пример использования  
if \_\_name\_\_ == "\_\_main\_\_":  
 filename = "seating\_chart.txt"  
  
 create\_seating\_chart(filename)  
  
 print(f"Количество свободных мест: {count\_free\_seats(filename)}")  
  
 row\_number = int(input("Введите номер ряда для проверки: "))  
 seat\_number = int(input("Введите номер места для проверки: "))  
  
 print(check\_seat(filename, row\_number, seat\_number))

**ex54.py:**

import json  
import os  
  
  
# Функция для загрузки телефонного справочника из файла  
def load\_phonebook(filename):  
 if not os.path.exists(filename):  
 return {}  
 with open(filename, 'r', encoding='utf-8') as file:  
 return json.load(file)  
  
  
# Функция для сохранения телефонного справочника в файл  
def save\_phonebook(filename, phonebook):  
 with open(filename, 'w', encoding='utf-8') as file:  
 json.dump(phonebook, file, ensure\_ascii=False, indent=4)  
  
  
# Функция для добавления нового контакта  
def add\_contact(phonebook, name, phone\_numbers):  
 if name in phonebook:  
 print(f"Контакт '{name}' уже существует. Добавляем номера к существующему контакту.")  
 phonebook[name].extend(phone\_numbers)  
 else:  
 phonebook[name] = phone\_numbers  
 print(f"Контакт '{name}' добавлен с номерами: {', '.join(phone\_numbers)}")  
  
  
# Функция для удаления контакта по имени  
def delete\_contact\_by\_name(phonebook, name):  
 if name in phonebook:  
 del phonebook[name]  
 print(f"Контакт '{name}' удален.")  
 else:  
 print(f"Контакт '{name}' не найден.")  
  
  
# Функция для поиска контакта по номеру телефона  
def search\_contact\_by\_phone(phonebook, phone\_number):  
 for name, numbers in phonebook.items():  
 if phone\_number in numbers:  
 return name, numbers  
 return None  
  
  
# Функция для удаления номера телефона из контактов  
def delete\_phone\_number(phonebook, name, phone\_number):  
 if name in phonebook:  
 if phone\_number in phonebook[name]:  
 phonebook[name].remove(phone\_number)  
 print(f"Номер '{phone\_number}' удален из контакта '{name}'.")  
 # Если больше нет номеров у контакта, можно удалить контакт  
 if not phonebook[name]:  
 del phonebook[name]  
 print(f"Контакт '{name}' удален, так как у него больше нет номеров.")  
 else:  
 print(f"Номер '{phone\_number}' не найден у контакта '{name}'.")  
 else:  
 print(f"Контакт '{name}' не найден.")  
  
  
# Основной код  
if \_\_name\_\_ == "\_\_main\_\_":  
 filename = 'phonebook.json'  
  
 # Загрузка телефонного справочника  
 phonebook = load\_phonebook(filename)  
  
 # Примеры использования функций  
 while True:  
 print("\nВыберите действие:")  
 print("1. Добавить контакт")  
 print("2. Удалить контакт по имени")  
 print("3. Поиск контакта по номеру телефона")  
 print("4. Удалить номер телефона из контактов")  
 print("5. Выход")  
  
 choice = input("Ваш выбор: ")  
  
 if choice == '1':  
 name = input("Введите имя контакта: ")  
 phone\_numbers = input("Введите номера телефонов через запятую: ").split(',')  
 # Убираем лишние пробелы вокруг номеров  
 phone\_numbers = [number.strip() for number in phone\_numbers]  
 add\_contact(phonebook, name, phone\_numbers)  
 save\_phonebook(filename, phonebook)  
  
 elif choice == '2':  
 name = input("Введите имя контакта для удаления: ")  
 delete\_contact\_by\_name(phonebook, name)  
 save\_phonebook(filename, phonebook)  
  
 elif choice == '3':  
 phone\_number = input("Введите номер телефона для поиска: ")  
 result = search\_contact\_by\_phone(phonebook, phone\_number)  
 if result:  
 name, numbers = result  
 print(f"Контакт найден: {name}, Номера: {', '.join(numbers)}")  
 else:  
 print("Контакт не найден.")  
  
 elif choice == '4':  
 name = input("Введите имя контакта: ")  
 phone\_number = input("Введите номер телефона для удаления: ")  
 delete\_phone\_number(phonebook, name, phone\_number)  
 save\_phonebook(filename, phonebook)  
  
 elif choice == '5':  
 break  
  
 else:  
 print("Некорректный выбор. Попробуйте снова.")

**Контрольные вопросы**

1. Что такое файл?

В контексте Python файл — это структура данных для хранения информации, с которой можно работать при помощи встроенных функций Python. Python предоставляет несколько встроенных функций для работы с файлами: чтение, запись, добавление данных, закрытие файла и т.д. Эти операции выполняются через файловый объект, который создается при помощи функции open(). Например, текстовый файл может хранить строки, которые потом можно прочитать или изменить через программу на Python.

1. Как определяют модуль pickle?

Модуль pickle в Python используется для сериализации и десериализации объектов, то есть для преобразования объектов Python (например, списков, словарей или даже пользовательских классов) в байтовую последовательность и наоборот. Сериализация позволяет сохранить сложные объекты в файл, чтобы позже их можно было восстановить в исходном виде. Модуль pickle подключается командой: import pickle

1. Что такое относительный путь? Как он задается?

Относительный путь — это путь к файлу или директории относительно текущего местоположения файла или программы. Относительный путь не указывает полный путь от корневой директории и обычно начинается с ./ (текущая директория) или ../ (родительская директория).

1. Что такое абсолютный путь? Как он задается?

Абсолютный путь — это полный путь к файлу или директории, начиная от корневой директории файловой системы. Абсолютный путь указывает местоположение файла независимо от текущей директории программы. Он обычно начинается с корня (/ для Unix-систем и C:\ для Windows).

1. Какие существуют форматы файлов и чем они отличаются?

В Python работа с файлами часто подразумевает чтение и запись данных в файлы различных форматов:

* **Текстовые файлы (.txt, .csv):** содержат текстовую информацию, строки или табличные данные в виде текста.
* **Бинарные файлы (.bin, .dat):** содержат двоичные данные, которые не читаются в текстовом виде; используются для хранения изображений, видео и сериализованных объектов (pickle).
* **JSON-файлы (.json):** текстовый формат для хранения данных в формате JavaScript Object Notation, используется для обмена данными между системами.
* **Excel-файлы (.xlsx, .xls):** формат для хранения таблиц, используется для обработки данных с помощью библиотек (pandas, openpyxl).
* **Форматы изображений (.jpg, .png):** содержат графические данные, читаются и обрабатываются специализированными библиотеками, такими как Pillow.

1. Функции открытия и закрытия файлов.

В Python файлы открываются функцией open() и закрываются функцией close() или автоматически через контекстный менеджер with. Контекстный менеджер with является предпочтительным способом работы с файлами в Python, так как он гарантирует автоматическое закрытие файла после использования, что предотвращает утечки памяти.

Функция open() принимает два обязательных аргумента: путь к файлу и режим:

* **'r':** чтение,
* **'w':** запись (перезапись существующего содержимого),
* **'a':** добавление данных в конец файла,
* **'rb', 'wb', 'ab':** работа с бинарными файлами.