**Задание на лабораторную работу №3**

**Тема**: Структуры данных. Ввод/вывод.

* Использовать пару стандартных структур данных согласно своему варианту.
* Продемонстрировать основные операции, которые можно применить к каждой из структур.
* Определить, какая из структур быстрее в операциях: доступа, добавления, удаления, объединения, пересечения, разницы, сравнения, поиска, сортировки и т.п., при условии, что обе структуры поддерживают данные операции.
* Обеспечить возможность сохранения данных в файл и загрузке из файла.
* Проверить работу программы.

Ход работы:  
import time  
import json  
  
# Функция для сохранения данных в файл  
def save\_data(filename, data):  
 with open(filename, 'w') as file:  
 if isinstance(data, dict):  
 json.dump(data, file)  
 elif isinstance(data, list):  
 json.dump(data, file)  
  
# Функция для загрузки данных из файла  
def load\_data(filename):  
 with open(filename, 'r') as file:  
 data = json.load(file)  
 return data  
  
# Сохранение данных в файлы  
list\_data = [1, 2, 3, 4, 5]  
dict\_data = {'name': 'Valery', 'surname': 'Frolov'}  
  
save\_data('list\_data.json', list\_data)  
save\_data('dict\_data.json', dict\_data)  
  
# Функция для проведения теста и сравнения времени выполнения операций над списком и словарем  
def compare\_operations(list\_data, dict\_data):  
 # Измерение времени для операций с list  
 list\_start\_time = time.time()  
 list\_operations(list\_data)  
 list\_end\_time = time.time()  
 list\_time = list\_end\_time - list\_start\_time  
  
 # Измерение времени для операций с dict  
 dict\_start\_time = time.time()  
 dict\_operations(dict\_data)  
 dict\_end\_time = time.time()  
 dict\_time = dict\_end\_time - dict\_start\_time  
  
 # Вывод результатов  
 print("\nВремя выполнения операций с list:", list\_time)  
 print("Время выполнения операций с dict:", dict\_time)  
 if list\_time < dict\_time:  
 print("Список быстрее словаря")  
 elif list\_time > dict\_time:  
 print("Словарь быстрее списка")  
 else:  
 print("Время выполнения операций с list и dict одинаково")  
  
# Операции с list  
def list\_operations(data):  
  
 data.append(6)  
 print("После добавления элемента в конец:", data)  
 data.remove(5)  
 print("После удаления элемента по значению 5:", data)  
  
 data2 = [7, 8, 9]  
 merged\_data = data + data2  
 print("Объединение двух списков:", merged\_data)  
  
 diff\_data = list(set(data) - set(data2))  
 print("Разница между списками:", diff\_data)  
 data.sort()  
 print("Сортировка списка:", data)  
  
# Операции с dict  
def dict\_operations(data):  
 data['age'] = 21  
 print("\nПосле добавления пары ключ-значение:", data)  
 del data['age']  
 print("После удаления пары ключ-значение:", data)  
 data2 = {'city': 'Cherepovets', 'country': 'Russia'}  
 merged\_data = {\*\*data, \*\*data2}  
 print("Объединение двух словарей:", merged\_data)  
 diff\_keys = data.keys() - data2.keys()  
 diff\_dict = {k: data[k] for k in diff\_keys}  
 print("Разница между ключами в двух словарях:", diff\_dict)  
  
 # Сортировка словаря по ключам  
 sorted\_keys = sorted(data.keys())  
 # Создание отсортированного словаря  
 sorted\_dict = {key: data[key] for key in sorted\_keys}  
 print("Отсортированный словарь:", sorted\_dict)  
  
# Вызов функции для проведения теста и сравнения времени выполнения операций  
compare\_operations(list\_data, dict\_data)

Результат работы:

