Лабораторная работа №3:

Реализовать параметризованный декоратор, который:

#По умолчанию пишет в консоль (sys.stdout)

#Может писать в текстовый файл в формате json (если передано имя файла '*.json') со следующей структурой одной записи: дата-время вызова функции, какая функция была вызвана и с какими параметрами и с каким результатом завершена.

#Может писать в базу данных sqlite3 (если передан объект типа sqlite3.Connection), развернутая в памяти компьютера

#”Рядом” с декоратором должна быть утилита, которая отображает содержимое базы данных с логированными данными

Код программы:

ЛР-3 > main.py > ...

```python
import sys
import functools
import sqlite3
import json
from datetime import datetime
from typing import Union, TextIO, Optional

def trace(func=None, *, handle=sys.stdout):
    if func is None:
        return lambda func: trace(func, handle=handle)

    @functools.wraps(func)
    def inner(*args, **kwargs):
        # Выполняем функцию и получаем результат
        result = func(*args, **kwargs)

        # Формируем запись лога
        log_entry = {
            'datetime': datetime.now().isoformat(),
            'func_name': func.__name__,
            'params': {
                'args': args,
                'kwargs': kwargs
            },
            'result': result
        }

        # Записываем лог в соответствующий обработчик
        if isinstance(handle, sqlite3.Connection):
            # Запись в SQLite
            try:
                cur = handle.cursor()
                cur.execute("""
                    INSERT INTO logtable (datetime, func_name, params, result)
                    VALUES (?, ?, ?, ?)
                """, (
                    log_entry['datetime'],
                    log_entry['func_name'],
                    json.dumps(log_entry['params']),
                    json.dumps(log_entry['result'])
                ))
```

```
                    handle.commit()
                except sqlite3.OperationalError:
                    # Если таблицы не существует, создаем ее
                    cur.execute("""
                        CREATE TABLE IF NOT EXISTS logtable (
                            id INTEGER PRIMARY KEY AUTOINCREMENT,
                            datetime TEXT,
                            func_name TEXT,
                            params TEXT,
                            result TEXT
                        )
                    """)
                    handle.commit()
                    # Повторяем попытку вставки
                    cur.execute("""
                        INSERT INTO logtable (datetime, func_name, params, result)
                        VALUES (?, ?, ?, ?)
                    """, (
                        log_entry['datetime'],
                        log_entry['func_name'],
                        json.dumps(log_entry['params']),
                        json.dumps(log_entry['result'])
                    ))
                    handle.commit()

            elif isinstance(handle, str) and handle.endswith('.json'):
                # Запись в JSON файл
                try:
                    with open(handle, 'r+') as f:
                        try:
                            logs = json.load(f)
                        except json.JSONDecodeError:
                            logs = []
                        logs.append(log_entry)
                        f.seek(0)
                        json.dump(logs, f, indent=2)
                except FileNotFoundError:
                    with open(handle, 'w') as f:
                        json.dump([log_entry], f, indent=2)
            else:
                # Запись в поток (по умолчанию sys.stdout)
                handle.write(f"\n{log_entry}\n")

            return result

        return inner

    def showlogs(con: sqlite3.Connection):
        """Утилита для отображения логов из SQLite базы"""
        try:
            cur = con.cursor()
            cur.execute("SELECT * FROM logtable")
            logs = cur.fetchall()

            if not logs:
                print("Логи отсутствуют")
                return

            print("\nЛоги из базы данных:")
            print("-" * 50)
            for log in logs:
                print(f"ID: {log[0]}")
                print(f"Дата/время: {log[1]}")
                print(f"Функция: {log[2]}")
                print(f"Параметры: {json.loads(log[3])}")
                print(f"Результат: {json.loads(log[4])}")
                print("-" * 50)
        except sqlite3.OperationalError:
            print("Таблица логов не существует")
```

```python
# Примеры использования:

@trace(handle=sys.stderr)
def increm(x):
    """Инкремент"""
    return x + 1

@trace(handle=sys.stdout)
def decrem(x):
    """Декремент"""
    return x - 1

# Вариант по умолчанию (консоль)
@trace
def f2(x):
    return x**2

# Запись в JSON файл
@trace(handle='logger.json')
def f3(x):
    return x**3

# Запись в SQLite базу
handle_for_f4 = sqlite3.connect(":memory:")

@trace(handle=handle_for_f4)
def f4(x):
    return x**4

# Тестирование
print(increm.__doc__)
increm(2)
decrem(2)
f2(5)
f3(3)
f4(4)

# Просмотр логов из базы
showlogs(handle_for_f4)
```

Результат:

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 2 семестр> & C:/Users/sambu/AppData/L
313/python.exe "c:/Users/sambu/OneDrive/Рабочий стол/Учеба в Герцена/II курс/Программирование Python, 2 курс, 2 семестр/ЛР-3/main.py"
Инкремент

{'datetime': '2025-06-12T20:49:30.304564', 'func_name': 'increm', 'params': {'args': (2,), 'kwargs': {}}, 'result': 3}

{'datetime': '2025-06-12T20:49:30.304700', 'func_name': 'decrem', 'params': {'args': (2,), 'kwargs': {}}, 'result': 1}

{'datetime': '2025-06-12T20:49:30.304791', 'func_name': 'f2', 'params': {'args': (5,), 'kwargs': {}}, 'result': 25}

Логи из базы данных:
--------------------------------------------------
ID: 1
Дата/время: 2025-06-12T20:49:30.306249
Функция: f4
Параметры: {'args': [4], 'kwargs': {}}
Результат: 256
```

Текстовый файл формата json: