Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

ЛАБОРАТОРНАЯ РАБОТА №1

По дисциплине Тестирование программного обеспечения

Тема работы Unit - тестирование

Обучающийся: Белисов Глеб Андреевич

Факультет ПИН

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

2025

Был выбран консольный Python-скрипт для учёта расходов со следующим функционалом:

- Чтение базы из файла (from_file_to_list).
- Добавление новой записи (zapis).
- Просмотр и фильтрация по дате и категории (data_sort, category, data_variable, category_variable).
- Сортировка по цене (cost_sort).
- Удаление и обновление записей (deleted, upd).
- Управление через главное меню (main_page).

Важные части проекта:

Парсинг данных из файла (from_file_to_list, upd) - без корректной загрузки и сохранения работа невозможна.

Добавление записей (zapis) важный участок, включающий валидацию данных.

Сортировка (cost_sort) требуется для корректного анализа расходов.

Группировка по категориям и датам (category, data_sort) очень важная функция для структурирования данных.

Удаление (deleted) важно для поддержания актуальности базы.

```python
def test_from_file_to_list(tmp_path):
    file_path = tmp_path / "test.txt"
    file_path.write_text("Milk 50 Food\nBread 30 Food\n", encoding="utf-8")
    result = from_file_to_list(str(file_path))
    assert result == [["Milk", "50", "Food"], ["Bread", "30", "Food"]]
```

Проверяет, что функция корректно читает данные из файла и преобразует строки в список.

```python
def test_minor_capitalization():
    massive = [["milk", "50", "food"], ["bread", "30", "food"]]
    result = minor(massive)
    assert result == [["Milk", "50", "Food"], ["Bread", "30", "Food"]]
```

Проверяет, что функция minor приводит все слова к виду с заглавной буквы.

```python
def test_category_grouping():
    massive = [["Milk", "50", "Food"], ["Chair", "100", "Furniture"], ["Bread", "30", "Food"]]
    result = category(massive)
    assert result == {"Food": ["Milk", "Bread"], "Furniture": ["Chair"]}
```

Проверяет работу функции category, которая группирует товары по категориям.

```python
def test_cost_sort_min(monkeypatch):
    massive = [["Milk", "50", "Food"], ["Bread", "30", "Food"], ["Chair", "100", "Furniture"]]
    monkeypatch.setattr("builtins.input", lambda _: "min")
    result = cost_sort(massive)
    assert [item[0] for item in result] == ["Bread", "Milk", "Chair"]
```

Проверяет, что сортировка по цене от меньшего к большему работает правильно.

```python
def test_data_sort_grouping():
    today = f"{datetime.datetime.now().day}-{datetime.datetime.now().month}-{datetime.datetime.now().year}"
    massive = [["Milk", "50", "Food", today], ["Bread", "30", "Food", today]]
    result = data_sort(massive)
    assert today in result
    assert "Milk" in result[today]
    assert "Bread" in result[today]
```

Проверяет, что функция data_sort корректно группирует товары по дате добавления.

```python
def test_zapis_invalid_price(monkeypatch):
    massive = []
    monkeypatch.setattr("builtins.input", lambda _: "Milk abc Food")
    with pytest.raises(ValueError, match="invalid literal for int"):
        zapis(massive)
```

Тестирует добавления записи zapis при некорректной цене

```
def test_deleted_remove(monkeypatch):
    massive = [["Milk", "50", "Food"]]
    monkeypatch.setattr("builtins.input", lambda _: "1")
    result = deleted(massive)
    assert result == []
```

Проверяет что функция deleted удаляет запись по правильному номеру.

```
def test_upd_write_and_read(tmp_path):
    file_path = tmp_path / "test.txt"
    massive = [["Milk", "50", "Food"], ["Bread", "30", "Food"]]
    upd(massive, str(file_path))
    read_back = from_file_to_list(str(file_path))
    assert read_back == massive
```

Проверяет, что функция upd сохраняет данные в файл.

Результаты

```
test_app.py::test_from_file_to_list PASSED                    [ 10%]
test_app.py::test_from_file_to_list PASSED                    [ 10%]
test_app.py::test_minor_capitalization PASSED                 [ 20%]
test_app.py::test_category_grouping PASSED                    [ 30%]
test_app.py::test_cost_sort_min PASSED                        [ 40%]
test_app.py::test_cost_sort_invalid PASSED                    [ 50%]
test_app.py::test_data_sort_grouping PASSED                   [ 60%]
test_app.py::test_zapis_add_valid PASSED                      [ 70%]
test_app.py::test_zapis_invalid_price PASSED                  [ 80%]
test_app.py::test_deleted_remove PASSED                       [ 90%]
test_app.py::test_upd_write_and_read PASSED                   [100%]
```

```
=============================== tests coverage ===============================
______________ coverage: platform win32, python 3.13.2-final-0 _______________

Name      Stmts   Miss  Cover   Missing
-------------------------------------
app.py      178     94    47%   31-44, 49-58, 71-72, 78-79, 91-94, 109-123, 135, 150-153, 156-157, 161-205
-------------------------------------
-------------------------------------
TOTAL       178     94    47%
```

Тесты покрывают 47% кода и все проходят успешно. Были применены AAA и FIRST, и тестирование подтвердило корректность работы в основных сценариях.