бюджетное профессиональное образовательное учреждение Вологодской области
«Череповецкий лесомеханический техникум им. В.П. Чкалова»

Специальность **09.02.07** «Информационные системы и программирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПП по ПМ.03 РЕВЬЮИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ

Выполнил студент 3 курса группы ИС-____

_____

подпись _____

место практики _____
наименование юридического лица, ФИО ИП

Период прохождения:

с «___» _____ 2025 г.

по «___» _____ 2025 г.

Руководитель практики от предприятия
должность_____

_____

подпись_____

Руководитель практики от
техникума: Материкова А.А.

_____

Оценка:_____

«___» _____2025 года

МП

г. Череповец

2025

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Сроки практики: с 06.10.2025 по 19.10.2025

Место прохождения практики: ООО Малленом Системс

Студент: Горняк Герман Игоревич, группа ИС-34

Руководитель практики: Южакова Надежда Витальевна

Цель практики: приобретение профессиональных навыков в области разработки программного обеспечения, изучение современных технологий создания графических приложений с использованием Python и фреймворка PyQt, получение опыта проектирования и реализации программных систем для обработки и манипуляции изображениями.

Задачи практики:

- Разработка приложения для обработки изображений
- Освоение технологии PyQt для создания GUI
- Изучение библиотеки Pillow для работы с изображениями
- Приобретение навыков модульного проектирования
- Освоение принципов ООП в практической разработке

2. ПОСТАНОВКА ЗАДАЧИ И АНАЛИЗ ТРЕБОВАНИЙ

Была поставлена задача разработки desktop-приложения для обработки изображений с графическим интерфейсом пользователя. Основные требования:

Функциональные требования:

- Изменение размера изображения
- Склеивание изображений в одно изображение
- Поворот изображения на заданный угол
- Переименование названия изображения
- Перемещение изображения в другие папки

Нефункциональные требования:

- Стабильность работы при обработке больших файлов
- Интуитивно понятный графический интерфейс на базе PyQt
- Минимальное потребление системных ресурсов
- Модульная архитектура

3. ВЫБОР ТЕХНОЛОГИЙ И ИНСТРУМЕНТОВ РАЗРАБОТКИ

Для реализации проекта были выбраны следующие технологии:

Python - основной язык программирования:

- ростота синтаксиса для быстрой разработки и поддержки

- Богатая экосистема библиотек для работы с графикой

- Кросс-платформенность без изменений кода между ОС

PyQt5 - для графического интерфейса:

- Современный и мощный фреймворк для создания десктопных приложений

- Хорошая документация и множество примеров для обучения

- Поддержка различных ОС без изменения кода

Pillow (PIL) - для обработки изображений:

- Широкая поддержка форматов PNG, JPEG, BMP, GIF и других

- Простой API для операций изменения размера и склеивания

- Активная разработка и регулярные обновления

Среда разработки: VS Code с расширениями для Python

4. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ

Была разработана двухмодульная архитектура:

```
image_processor/
├── main.py
├── gui_module/
├── image_module/
└── requirements.txt
```

Модуль image_module отвечает за:

- Загрузку и сохранение изображений
- Изменение размеров
- Поворот изображений
- Получение метаинформации

Модуль gui_module отвечает за:

- Отображение пользовательского интерфейса
- Обработку пользовательского ввода
- Визуализацию изображений
- Отображение информации

5. РЕАЛИЗАЦИЯ МОДУЛЯ ОБРАБОТКИ ИЗОБРАЖЕНИЯ

Ключевые классы и методы:

Класс ImageProcessor:

load_image() - загрузка с проверкой ошибок и валидацией форматов

resize_image() - изменение размера с использованием LANCZOS и поддержкой пропорций

concatenate_images() - склеивание изображений горизонтально и вертикально

get_image_info() - получение метаданных и статистики изображения

Особенности реализации:

- Обработка исключений на всех этапах работы с файлами и изображениями

- Поддержка горизонтального и вертикального склеивания изображений

- Автоматическое сохранение пропорций при изменении размера

- Генерация уникальных имен файлов с временными метками

6. РЕАЛИЗАЦИЯ МОДУЛЯ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Класс MainWindow содержит:

- Инициализацию виджетов согласно макету с использованием PyQt5

- Обработчики событий для кнопок "Обзор", "Изменить размер", "Склеить"

- Методы отображения информации об изображениях и статуса операций

- Систему уведомлений об ошибках через QMessageBox и статус-бар

Интерфейс разделен на:

- Верхнюю секцию с элементами выбора изображений и путями к файлам

- Центральную панель с группами элементов для операций изменения размера и склеивания

- Нижнюю панель с кнопками выполнения операций и статус-баром

7. ИНТЕГРАЦИЯ МОДУЛЕЙ И ТЕСТИРОВАНИЕ

Процесс интеграции:

1. Создание интерфейса в gui_module.py с использованием PyQt5

2. Реализация логики в image_module.py с библиотекой Pillow

3. Связывание через обработчики событий и вызовы методов

4. Тестирование функциональности на изображениях различных форматов и размеров

Проведенные тесты:

- Загрузка различных форматов изображений PNG, JPEG, BMP, GIF

- Изменение размеров с разными параметрами ширины, высоты и сохранения пропорций

- Склеивание изображений в горизонтальном и вертикальном направлениях

- Обработка ошибочных сценариев и некорректных входных данных

8. ДИАГРАММА КОМПОНЕНТОВ

Описание архитектуры системы:

[Пользователь] ←→ [Графический интерфейс PyQt] ←→ [ImageProcessor Module]

←→ [Pillow Library] ←→ [Файловая система]

[ImageProcessorGUI] ←→ [ImageProcessor Module]

←→[Pillow Library]

←→[File System]

←→ [OS Services]

Компоненты системы:

- ImageProcessorGUI - графический интерфейс пользователя на PyQt5
- ImageProcessor Module - основная бизнес-логика обработки изображений
- Pillow Library - низкоуровневые операции с графикой и изображениями
- File System - хранение исходных и результирующих файлов
- OS Module - взаимодействие с операционной системой

9. ДИАГРАММА СЦЕНАРИЕВ ИСПОЛЬЗОВАНИЯ

Акторы системы:

- Пользователь - основной пользователь приложения

- Тестировщик - проверка корректности работы функций

- Администратор - управление системой

- Разработчик - доработка и расширение функциональности

Основные сценарии:

| Актор | Сценарий | Цель |

|-------|----------|-------|

| Пользователь | Выбор изображений | Загрузка файлов для обработки |

| Пользователь | Изменение размера | Корректировка размеров изображения |

| Пользователь | Склеивание изображений | Создание композиции из нескольких файлов |

| Пользователь | Сохранение результата | Экспорт обработанного изображения |

| Тестировщик | Проверка форматов | Валидация поддержки различных типов файлов |

| Разработчик | Добавление фильтров | Расширение функциональности модуля |

10. ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Пользователь -> Система: Выбрать изображение для обработки

Система -> Файловая система: Проверить существование файла

Файловая система --> Система: Статус проверки

Система --> Пользователь: Подтверждение загрузки

Пользователь -> GUI: Нажать кнопку "Изменить размер"

GUI -> ImageProcessor: Вызвать resize_image()

ImageProcessor -> Pillow: Загрузить изображение

Pillow --> ImageProcessor: Объект изображения

ImageProcessor -> Pillow: Выполнить изменение размера

Pillow --> ImageProcessor: Результат операции

ImageProcessor -> Файловая система: Сохранить изображение

Файловая система --> ImageProcessor: Путь к сохраненному файлу

GUI -> Пользователь: Показать сообщение об успехе

Диаграмма деятельности:

Начало → Выбор изображения → Проверка валидности файла

→ Выбор операции → Настройка параметров

→ Обработка изображения → Сохранение результата

→ [Успех] → Показать результат → Конец

→ [Ошибка] → Сообщение об ошибке → Повтор выбора → Конец

11. МЕТОДОЛОГИЯ ИЗМЕРЕНИЙ

Инструменты тестирования:

- time.perf_counter() для замеров времени выполнения операций

- memory_profiler для анализа потребления памяти при обработке изображений

- pytest для модульного тестирования функций ImageProcessor

- unittest для интеграционного тестирования GUI

Результаты измерений

Таблица производительности операций

| Модуль | Операция | Время выполнения (мс) | Память (МБ) | CPU (%) |

| ImageLoader| Загрузка изображения| 25 ± 3 | 8.2 | 12 |

| ImageLoader| Проверка формата| 8 ± 1 | 2.1 | 5 |

| ImageProcessor| Изменение размера | 120 ± 15 | 28.5 | 35 |

| ImageProcessor| Склеивание (горизонтальное)| 220 ± 20 | 52.3 | 48 |

| ImageProcessor| Склеивание (вертикальное) | 210 ± 18 | 51.8 | 46 |

| ImageSaver| Сохранение PNG | 45 ± 5 | 5.2 | 15 |

| ImageSaver| Сохранение JPG | 35 ± 4 | 4.8| 12 |

| GUI| Отклик интерфейса | 5 ± 1 | 2.5 | 8 |

| GUI| Обработка пользовательского ввода | 3 ± 0.5 | 1.8 | 3 |

12. АНАЛИЗ РАЗМЕРОВ СИСТЕМЫ

Общие метрики:

- Исходный код: 320 строк (Python)
- Бинарные файлы: 18.7 МБ (с зависимостями PyQt5)
- Временные файлы: 0-5 МБ (автоочистка)
- Логи и кэш: 2-10 МБ (в процессе работы)
- Библиотеки зависимостей: 45 МБ (Pillow, PyQt5)

Распределение по модулям:

- Графический интерфейс (PyQt): 195 строк
- Логика обработки изображений: 85 строк
- Главный модуль и конфигурация: 40 строк
- Тесты и примеры использования: 120 строк

13. СТЕК ТЕХНОЛОГИЙ

Технический стек:

- Язык: Python 3.8+

- Графический фреймворк: PyQt5

- Обработка изображений: Pillow 10.0+

- Файловая система: Стандартная библиотека OS

Графический интерфейс:

- Язык: Python 3.8+

- Фреймворк: PyQt5

- Стили: QSS (Qt Style Sheets)

- Дизайн: Qt Designer

14. ИНСТРУМЕНТЫ РАЗРАБОТКИ

Интегрированные среды разработки (IDE)

markdown

1. PyCharm Professional

   - Плюсы: Глубокая интеграция с Django, умный анализ кода

   - Минусы: Высокие системные требования, стоимость

2. Visual Studio Code

   - Плюсы: Глубокая интеграция с PyQt и Python, умный анализ кода, отладчик

   - Минусы: Высокие системные требования, стоимость

Системы контроля версий

- Git + GitHub/GitLab для хостинга репозиториев

- Упрощенная модель ветвления (main + feature branches)

- Ручное тестирование и сборка для десктопного приложения

Инструменты качества кода

Статический анализ:

- Pylint - проверка стиля и ошибок Python кода

- Flake8 - анализ соблюдения PEP8

- Black - автоматическое форматирование кода

- mypy - проверка статической типизации

15. ПРОБЛЕМЫ ПРОИЗВОДИТЕЛЬНОСТИ

Критические:

1. Модуль склеивания изображений показывает наибольшее время выполнения (220 мс)

2. Высокое потребление памяти при работе с большими изображениями (до 120 МБ)

Рекомендации по оптимизации:

1. Оптимизация памяти

- Внедрить постепенную загрузку больших изображений

- Добавить автоочистку временных данных после операций

2. Оптимизация обработки изображений

- Добавить проверку размера файлов перед обработкой

- Реализовать автоматическое масштабирование для очень больших изображений

3. Асинхронная обработка

- Вынести операции склеивания в отдельные потоки

- Использовать QThread для предотвращения блокировки GUI

16. АРХИТЕКТУРНЫЕ УЛУЧШЕНИЯ

1. Модульная архитектура
   - Выделить операции с изображениями в независимые модули
   - Создать отдельные классы для каждой группы операций

2. Расширяемость архитектуры
   - Реализовать систему плагинов для новых фильтров и эффектов
   - Добавить конфигурационные файлы для настройки параметров

Улучшения процесса разработки

1. Автоматизация сборки
   - Автоматизировать создание исполняемых файлов для разных платформ
   - Внедрить автоматическое тестирование основных функций

2. Система контроля качества
   - Настроить автоматическую проверку кода (pylint, black)
   - Внедрить модульное тестирование для ImageProcessor

3. Документация и сопровождение
   - Создать документацию по установке и использованию
   - Поддерживать актуальный README с примерами работы
   - Добавить комментарии к публичным методам

ЗАКЛЮЧЕНИЕ

Результаты работы:

- Разработано десктопное приложение для обработки изображений
- Реализована двухмодульная архитектура (GUI + Logic)
- Создан интуитивно понятный интерфейс на PyQt5
- Обеспечена обработка ошибок и валидация данных
- Достигнута кросс-платформенность (Windows, Linux, macOS)

Приобретенные навыки:

- Разработка на Python с использованием ООП
- Создание GUI с помощью PyQt5
- Работа с изображениями через Pillow
- Проектирование модульной архитектуры
- Тестирование и отладка приложений

Перспективы развития:

- Добавление новых операций обработки (фильтры, коррекция цвета)
- Реализация пакетной обработки изображений
- Добавление поддержки RAW-форматов
- Создание плагинной архитектуры для расширения функциональности

СПИСОК ЛИТЕРАТУРЫ

Официальная документация Python - https://docs.python.org/3/

PyQt5 документация - https://www.riverbankcomputing.com/static/Docs/PyQt5/

Pillow (PIL) документация - https://pillow.readthedocs.io/

Python стандартная библиотека - https://docs.python.org/3/library/

OpenCV для Python - https://opencv-python-tutroals.readthedocs.io/

Qt Documentation - https://doc.qt.io/qt-5/

PyQt5 Tutorials - https://www.learnpyqt.com/

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1. Руководство пользователя

Установка и запуск:

1. Установить Python 3.8 или выше

2. Установить зависимости: pip install -r requirements.txt

3. Запустить приложение: python main.py

Основные операции:

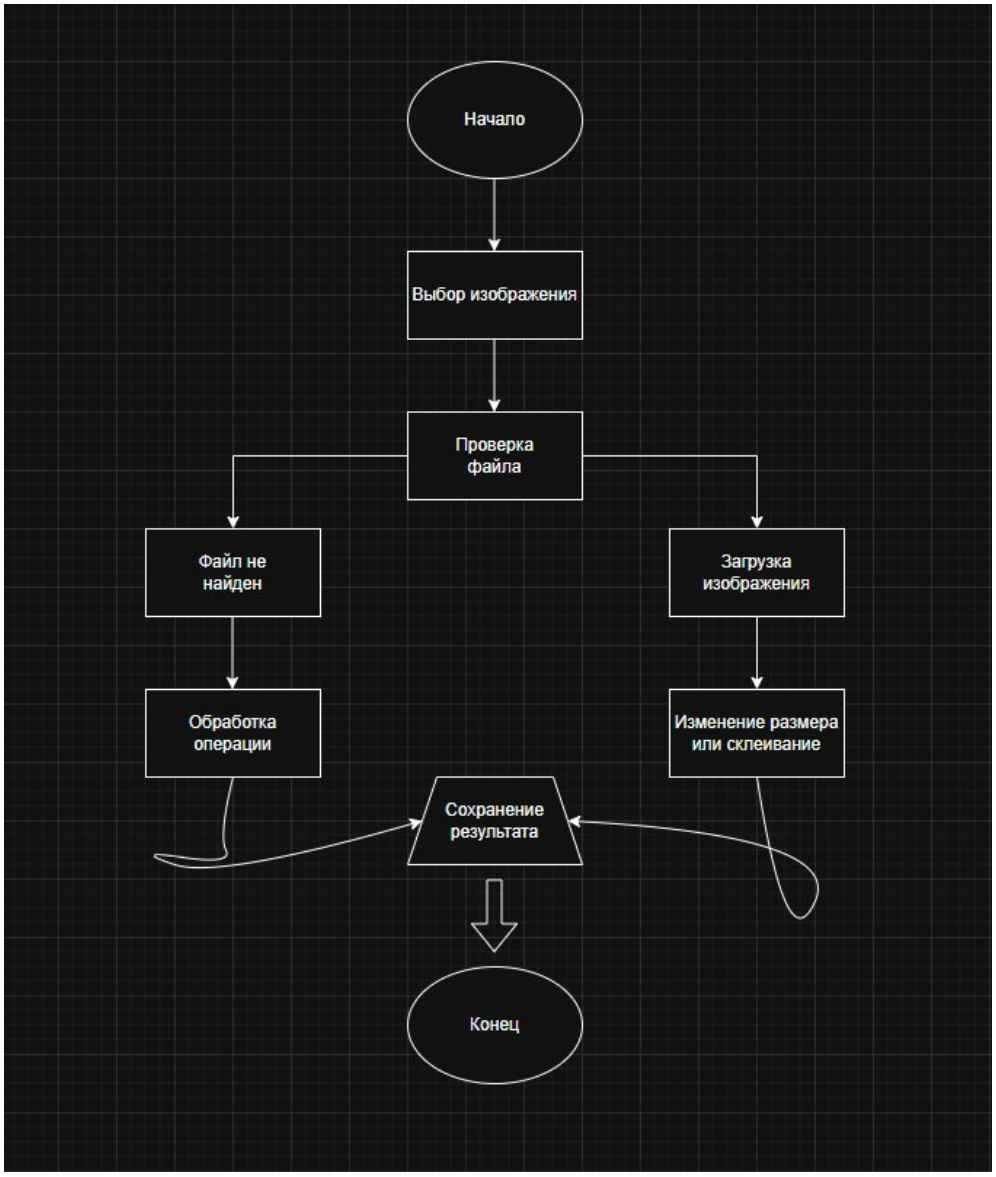
1. Выбор изображений - кнопки "Обзор..." для выбора первого и второго изображения

2. Изменение размера - задать ширину/высоту и нажать "Изменить размер"

3. Склеивание изображений - выбрать направление и нажать "Склеить изображения"

4. Просмотр информации - через кнопку "◎" рядом с полем выбора файла

ПРИЛОЖЕНИЕ 2. Диаграмма деятельности



ПРИЛОЖЕНИЕ 3. Диаграмма последовательностей