

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

КУРСОВАЯ РАБОТА
ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

доцент, канд. техн. наук
должность, уч. степень, звание

подпись, дата

С.А. Чернышев
инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

SORT YOUR PHOTOS

по дисциплине:
ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТЫ гр. №

4217

4217

подпись, дата

Д.М. Никитин

А.Н. Медянкина
инициалы, фамилия

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1. Анализ и описание предметной области	4
2. Выбор стека используемых технологий	6
3. Структурная схема архитектуры разрабатываемого приложения	7
4. Разработка приложения.....	9
5. Демонстрация работы приложения.....	19
ЗАКЛЮЧЕНИЕ.....	24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	25
ПРИЛОЖЕНИЕ А. Техническое задание на разрабатываемое программное обеспечение.....	26
ПРИЛОЖЕНИЕ Б. Методика приемо-сдаточных испытаний.....	30
ПРИЛОЖЕНИЕ В. Исходный код	34

ВВЕДЕНИЕ

В современном мире цифровая фотография играет важную роль, а количество изображений, сохраняемых на смартфонах, компьютерах и других устройствах, продолжает стремительно расти. Однако управление этим огромным массивом фотографий часто вызывает сложности. В частности, пользователи сталкиваются с проблемой поиска и систематизации снимков, особенно если они сделаны на разных устройствах или в разное время.

Данная курсовая работа посвящена разработке приложения для обработки и сортировки фотографий, целью которого является упрощение процесса организации фотоархивов. Программа предоставляет пользователям возможность сортировать фотографии по дате их создания или названию устройства, на которое они были сделаны. Уникальность приложения заключается в интуитивно понятном интерфейсе, доступных настройках и высокопроизводительном алгоритме обработки данных.

Актуальность данного приложения обусловлена его уникальностью. Всевозможные аналоги, по типу Tonfotos или DigiKam, не оснащены возможностью сортировать по устройству создания файла (фотографии). Так же приложение «sort your photos» доступно в opensource, так что каждый сможет его себе установить и не будет риска кражи данных.

Приложение ориентировано на широкий круг пользователей, включая как профессионалов, так и обычных владельцев смартфонов, желающих навести порядок в своих цифровых архивах. Используемый стек технологий, включающий язык программирования C++, фреймворк Qt, Exiv2, OpenSSL, обеспечивают стабильную работу, удобство интерфейса и возможность дальнейшего расширения функционала.

1. Анализ и описание предметной области

Приложение для сортировки фотографий по дате или названию телефона решает задачу организации большого количества изображений, которые могут накапливаться у пользователей. В современном мире смартфоны стали основным инструментом для фотографирования, и у многих возникает проблема поиска нужного снимка среди множества файлов. Сортировка на основе даты помогает упорядочить фотографии в хронологическом порядке, что удобно, например, для создания альбомов по событиям или воспоминаниям. Сортировка по названию телефона особенно полезна, если у пользователя есть изображения, сделанные на разных устройствах, и нужно разделить их для удобства работы или анализа.

Предполагается, что основными пользователями приложения будут те, кто регулярно фотографирует и хранит снимки на компьютере, а также те, кто ищет простой способ систематизировать свои коллекции. Приложение предоставляет интуитивно понятный интерфейс, где пользователь выбирает папку с исходными фотографиями и указывает, куда сохранить результат сортировки. Доступность выбора критерия сортировки — по дате или по названию устройства — делает программу универсальной и адаптируемой под разные задачи.

Приложение учитывает, что процесс сортировки может занимать время, и информирует об этом пользователя. Визуальная составляющая интерфейса, включая стильные градиенты и аккуратно оформленные элементы, делает использование приятным и удобным. Сама работа с фотографиями предполагает анализ их метаданных, что отражает важность встроенной функциональности для взаимодействия с файлами. Такое приложение полезно как для обычных пользователей, так и для профессионалов, которым важно поддерживать порядок в своих фотоархивах.

Программа ориентирована на управление папками с фотографиями, предоставляя пользователю возможность выбрать директорию для сохранения или сортировки изображений. Пользователи могут выбирать папку, где хранятся фотографии, и выбирать метод сортировки изображений, основываясь на критериях, таких как название устройства (например, телефона) или дата создания фотографии.

Предметная область данной программы лежит в области управления файлами, а именно в упрощении процессов сортировки и организации фотографий. Это востребовано среди пользователей, которые работают с большим количеством цифровых фотографий (как, например, фотографов, дизайнеров или обычных пользователей с накопившимися за время перемешанными вразнобой фотографиями). Сортировка по устройству или дате облегчает доступ к нужным фотографиям и упрощает организацию. Так что основная задача, которую решает эта программа — облегчение процесса сортировки и управления файлами фотографий.

2. Выбор стека используемых технологий

Для разработки приложения было выбрано использование C++ в сочетании с фреймворком Qt. Такой выбор технологий обоснован их производительностью, мощными возможностями работы с пользовательским интерфейсом и кроссплатформенностью. Также за счет того, что этот стек технологий пользуется популярностью среди разработчиков, будет проще найти к нему информацию.

C++ был выбран как основной язык программирования благодаря его эффективности и гибкости. Он обеспечивает низкоуровневый контроль над ресурсами, а также предоставляет возможность разделять обработку больших данных на несколько потоков, что особенно важно для работы с файловой системой и обработки огромных коллекции фотографий.

Qt предоставляет обширную библиотеку инструментов для создания графических интерфейсов, включая виджеты, стили, управление макетами и взаимодействие с пользователем. Это позволяет разработать интуитивно понятный и визуально привлекательный интерфейс, который отвечает современным требованиям удобства и эстетики. Кроме того, Qt предоставляет готовые решения для работы с файловыми диалогами, кнопками, радиокнопками, текстовыми полями и другими элементами интерфейса, что ускоряет процесс разработки. Также стоит отметить, что Qt поддерживает гибкое стилизование элементов интерфейса через стили и CSS-подобные правила, что позволило создать современный дизайн приложения с использованием градиентов и цветовых схем.

Использование C++ в связке с Qt также позволяет легко работать с метаданными изображений, обеспечивая высокую производительность при сортировке и анализе файлов. Таким образом, использование C++ и Qt сделало возможным создание производительного, стабильного и удобного приложения с широкими возможностями для дальнейшего расширения функционала.

3. Структурная схема архитектуры разрабатываемого приложения

Структурная схема архитектуры разрабатываемого приложения организована так, чтобы обеспечить читаемость кода и возможность масштабирования функционала. Архитектура приложения разделена на несколько основных компонентов, каждый из которых отвечает за определенную функциональность.

Главное окно (MainWindow) служит центральной точкой взаимодействия пользователя с приложением. Оно отвечает за ввод исходных данных (выбор папки с фотографиями и папки для сохранения отсортированных файлов), а также за выбор метода сортировки (по дате или названию телефона). Основные элементы интерфейса в главном окне включают текстовые поля, кнопки для открытия диалогов выбора папок, радиокнопки для выбора метода сортировки и кнопку для запуска процесса. Этот модуль также управляет переходом к другим частям приложения.

Окно загрузки (LoadingWidget) отображает информацию о ходе выполнения процесса сортировки. Этот компонент создается и активируется после запуска сортировки, предоставляя пользователю обратную связь о состоянии обработки данных. Использование отдельного окна для загрузки позволяет разгрузить главное окно и сосредоточиться только на отображении прогресса выполнения задачи.

Затем в том же окне выполняется основная логика сортировки файлов. Оно обрабатывает фотографии, анализируя их метаданные, такие как дата создания или информация о телефоне, с помощью встроенных библиотек C++ и Qt. После сортировки фотографии распределяются по указанным папкам и отображаются в ранее выбранной папке для отображения сортировки.

Модуль обработки фотографий отвечает за анализ и извлечение метаданных изображений. Это ключевая часть приложения, которая обеспечивает сортировку на основе выбранного пользователем критерия. Используется функциональность Qt для работы с файлами и метаданной.

Архитектура приложения построена таким образом, чтобы каждый модуль

был изолирован и отвечал за свою задачу. Это облегчает отладку, тестирование и модификацию отдельных частей программы. Главный фокус сделан на простоте взаимодействия пользователя с интерфейсом и надежности выполнения задач сортировки фотографий.

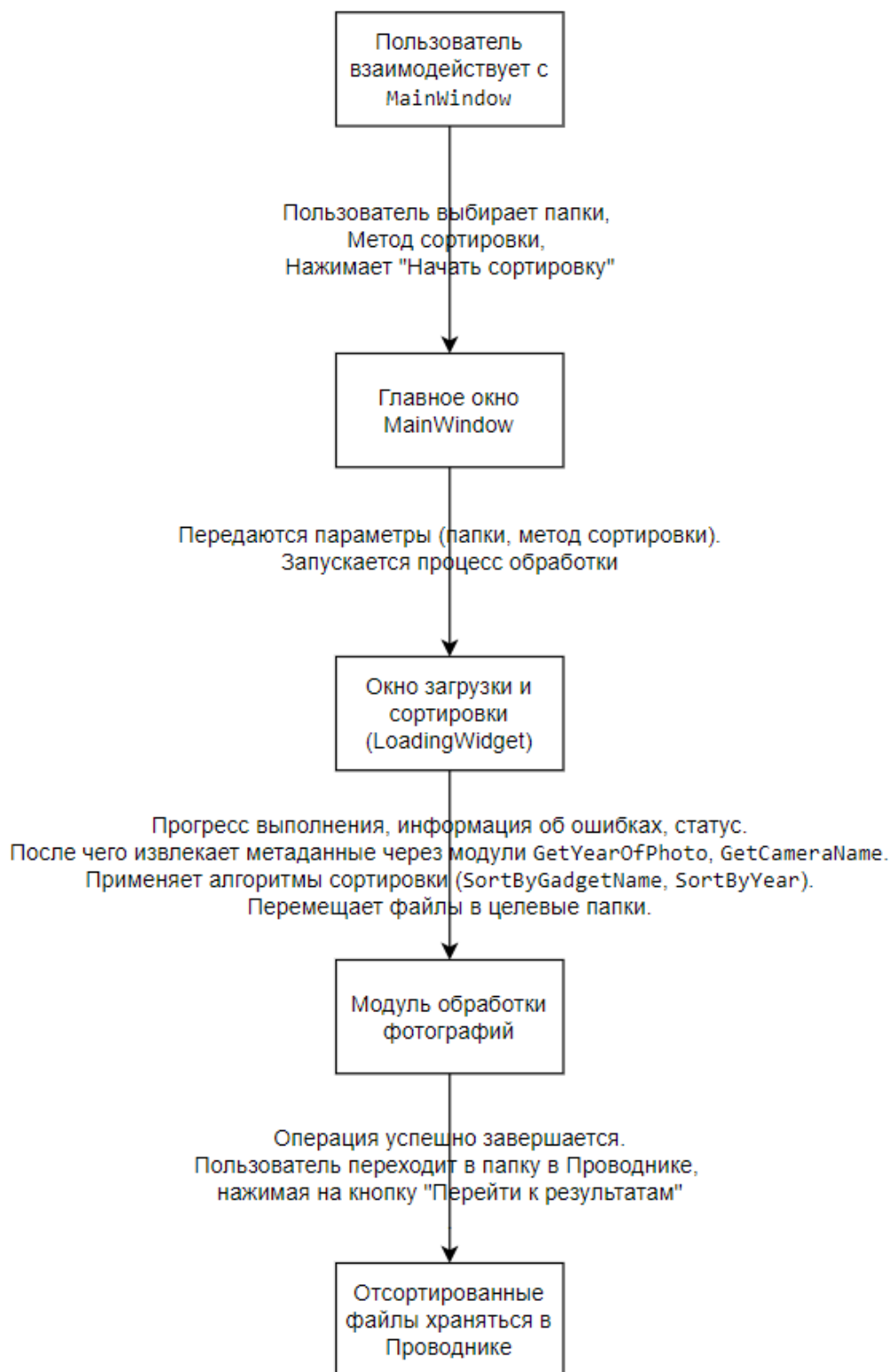


Рисунок 1 - Схема архитектуры приложения

4. Разработка приложения

Данное приложение состояло из 9 файлов .cpp и 8 файлов .h, написанных на C++ и 3 файлов .cpp, 3 файлов .h, написанных на фреймворке Qt Creator.

Вся основная часть бэкенда разрабатывалась в папке ProgramFiles. Основная задача была в том, чтобы программа определяла дату создания файла и устройство, на которое оно было сделано, а также умела распознавать дубликаты и могла заносить их в отдельную папку. Этого удалось добиться при помощи следующих наработок.

1. Работа с хешами файлов.

Код, фрагмент которого представлен в Листинге 1, вычисляет криптографический хэш файла с использованием алгоритма SHA-256. Хэш представляет собой уникальную строку фиксированной длины, которая однозначно идентифицирует содержимое файла. Для этого файл читается блоками данных (по 8192 байта), и каждый блок передаётся в криптографическую функцию обработки. Если чтение файла или этапы хэширования завершаются с ошибкой, программа либо возвращает пустую строку, либо выбрасывает исключение. После того как весь файл обработан, вычисленный хэш преобразуется в строку шестнадцатеричного формата, которая возвращается как результат. Таким образом, данный код позволяет проверить целостность файла или использовать хэш как его уникальный идентификатор.

Листинг 1. Добавление «computeFileHash.cpp»

```
#include "computeFileHash.h"
std::string computeFileHash(const fs::path& filePath) {
    std::ifstream file(filePath, std::ios::binary);
    if (!file) {
        return "";
    }
    EVP_MD_CTX* ctx = EVP_MD_CTX_new();
    if (!ctx) {
        throw std::runtime_error("Failed to create OpenSSL context.");
    }
    const EVP_MD* md = EVP_sha256();
    if (!md) {
        EVP_MD_CTX_free(ctx);
        throw std::runtime_error("Failed to initialize SHA256.");
    }
}
```

```

if (EVP_DigestInit_ex(ctx, md, nullptr) != 1) {
    EVP_MD_CTX_free(ctx);
    throw std::runtime_error("Failed to initialize Digest.");
}

```

2. Проверка файла на уникальность.

Следующий код из Листинга 2 принимает путь к файлу и возвращает уникальный путь, гарантируя, что файл с таким именем не существует. Если файл с указанным именем уже существует, к имени файла добавляется числовой суффикс (например, `_1`, `_2` и так далее) до тех пор, пока не будет найдено уникальное имя. Итоговый уникальный путь возвращается в виде объекта `fs::path`.

Листинг 2. Добавление «getUniquePath.cpp»

```

#include "getUniquePath.h"

fs::path getUniquePath(const fs::path& originalPath) {
    fs::path uniquePath = originalPath;
    int counter = 1;

    // Пока файл с таким именем существует, добавляем к имени суффикс с номером
    while (fs::exists(uniquePath)) {
        uniquePath = originalPath.parent_path() /
            (originalPath.stem().string() + "_" + std::to_string(counter) +
            originalPath.extension().string());
        counter++;
    }

    return uniquePath;
}

```

3. Проверка на наличие признаков изображения.

Код ниже проверяет, является ли указанный файл изображением, основываясь на его расширении. Он сравнивает расширение файла с заранее определённым списком популярных форматов изображений (таких как `.jpg`, `.png`, `.bmp` и другие). Перед сравнением расширение приводится к нижнему регистру, чтобы избежать ошибок из-за разницы в написании. Если расширение файла совпадает с одним из списка, функция возвращает `true`, иначе — `false`.

Листинг 3. Добавление «IsImageFile.cpp»

```

#include "IsImageFile.h"
bool isImageFile(const fs::path& filePath) {

```

```

// Для упрощения проверки считаем, что изображения имеют
расширения .jpg, .jpeg, .png, .bmp и т. д.
std::vector<std::string> imageExtensions = { ".jpg", ".jpeg", ".png",
".bmp", ".tiff", ".gif", ".dng" };

std::string extension = filePath.extension().string();
std::transform(extension.begin(), extension.end(),
extension.begin(), ::tolower); // Приводим расширение к нижнему регистру

return std::find(imageExtensions.begin(), imageExtensions.end(),
extension) != imageExtensions.end();
}

```

4. Приведение к общему виду и удаление недопустимых символов.

Этот код удаляет недопустимые символы из имени папки, заменяя их на подчеркивания. Он проверяет каждый символ в строке имени папки и, если символ входит в список недопустимых символов (например, \, /, *, ? и другие, которые нельзя использовать в именах файлов и папок в Windows), заменяет его на подчеркивание. В результате получается строка, подходящая для использования в качестве имени папки.

Листинг 4. Добавление «SanitizeFolderName»

```

#include "SanitizeFolderName.h"
// Функция для удаления недопустимых символов из имени папки
std::string sanitizeFolderName(const std::string& name) {
    std::string sanitized = name;
    // Список символов, которые нельзя использовать в именах файлов/папок в
Windows
    std::string invalidChars = "\\/:*?\"<>|";

    // Заменяем все недопустимые символы на подчеркивания
    for (char& c : sanitized) {
        if (invalidChars.find(c) != std::string::npos) {
            c = '_'; // Заменяем недопустимый символ на _
        }
    }
    return sanitized;
}

```

Далее переходим к папке по работе с передачей параметров фото, чтобы потом отсортировать их по названию камеры.

5. Получение названия устройства, на которое было сделано фото.

Следующий код извлекает название камеры, с помощью которой было сделано изображение, из метаданных EXIF файла. Он открывает изображение с

использованием библиотеки Exiv2, считывает его метаданные и пытается получить значения полей Exif.Image.Make (производитель камеры) и Exif.Image.Model (модель камеры). Затем объединяет эти строки и возвращает их в формате "Производитель Модель" (например, Canon EOS 5D). Если в изображении отсутствуют EXIF-данные, функция возвращает NoExifData. Если при обработке возникает ошибка, возвращается строка с описанием ошибки.

Листинг 5. Добавление «GetCameraName.cpp»

```
#include "GetCameraName.h"
#include <exiv2/exiv2.hpp>
std::string GetCameraName(const std::string& imagePath) {
    try {
        std::unique_ptr<Exiv2::Image> image =
Exiv2::ImageFactory::open(imagePath);
        image->readMetadata();

        Exiv2::ExifData& exifData = image->exifData();
        if (exifData.empty()) {
            return "NoExifData";
        }
        std::string make = exifData["Exif.Image.Make"].toString();
        std::string model = exifData["Exif.Image.Model"].toString();

        // Объединяем строку
        std::string MakedModel = make + " " + model;

        return MakedModel;
    }
    catch (Exiv2::Error& e) {
        return "Error " + std::string(e.what());
    }
}
```

6. Определение даты создания фото.

Данный код извлекает год, в который была сделана фотография, из EXIF-данных изображения. Он открывает изображение с помощью библиотеки Exiv2, считывает его метаданные и ищет ключ Exif.Photo.DateTimeOriginal, который содержит дату и время съемки. Если ключ найден, из строки даты извлекаются первые четыре символа, представляющие год (например, «2023»). Если EXIF-данные отсутствуют или в них нет нужного ключа, возвращается строка NoYearData. В случае ошибки обработки изображения возвращается сообщение об ошибке с её описанием.

Листинг 6. Добавление «GetYearOfPhoto.cpp»

```
#include "GetYearOfPhoto.h"
std::string GetYearOfPhoto(const std::string& imagePath) {
    try {
        std::unique_ptr<Exiv2::Image> image =
Exiv2::ImageFactory::open(imagePath);
        image->readMetadata();

        Exiv2::ExifData& exifData = image->exifData();
        if (exifData.empty()) {
            return "NoYearData";
        }

        // Проверяем наличие тега даты и времени съемки
        if (exifData.findKey(Exiv2::ExifKey("Exif.Photo.DateTimeOriginal")) !=
exifData.end()) {
            std::string dateTimeOriginal =
exifData["Exif.Photo.DateTimeOriginal"].toString();

            // Извлекаем год (формат "YYYY:MM:DD HH:MM:SS")
            if (dateTimeOriginal.length() >= 4) {
                return dateTimeOriginal.substr(0, 4); // Первые четыре символа –
это год
            }
        }
        return "NoYearData";
    }
    catch (Exiv2::Error& e) {
        return "Error " + std::string(e.what());
    }
}
```

Далее мы переходим к папке SortPhotosAlgorithms, в которой и производятся все операции по разработке всех алгоритмов сортировки фото и привязки их к проводнику.

7. Разработка алгоритма для сортировки фото на основе информации о камере.

Этот код будет уже пообъемнее и в написании, и в описании, ведь он предназначен для сортировки фотографий из указанной директории на основе информации о камере, которая была использована для их создания. Программа проходит по всем файлам в исходной папке (и её подкаталогам, если вызывается рекурсивная версия) и определяет, является ли файл изображением. Если это изображение, из его EXIF-данных извлекается имя камеры. Затем фотография перемещается в соответствующую папку, названную в честь камеры.

Если у изображения отсутствуют EXIF-данные или они повреждены, оно перемещается в папку NoExifData. Если файл не является изображением, он перемещается в папку NotPhotos. Код также проверяет, является ли файл дубликатом, используя хеш-сумму файла. Дубликаты помещаются в отдельную папку Duplicates внутри соответствующих категорий (камера, файлы без EXIF-данных или не-изображения).

Так же важно подметить, что для ускорения обработки используется многопоточность: файлы разбиваются на части, каждая из которых обрабатывается в отдельном потоке. В результате обеспечивается быстрая сортировка большого количества файлов. Программа учитывает возможные ошибки, такие как проблемы с доступом к файлам, и логирует их для диагностики.

Листинг 7. Добавление «SortByGadgetName.cpp»

```
#include "SortByGadgetName.h"
#include "../CommonFunctions/IsImageFile.h"
#include "../CommonFunctions/SanitizeFolderName.h"
#include "../CommonFunctions/computeFileHash.h"
#include "../CommonFunctions/getUniquePath.h"
#include "../GetPhotosParameters/GetCameraName.h"
namespace fs = std::filesystem;

// Мьютекс для синхронизации вывода (если нужно логировать работу потоков)
void processFilesWithDevice(
    const std::vector<fs::path>& files,
    const fs::path& targetDirectory,
    std::unordered_map<std::string, int>& hashMap
) {
    std::mutex coutMutex;
    static std::mutex hashMapMutex; // Мьютекс для защиты hashMap
    for (const auto& filePath : files) {
        try {
            // Считаем хеш файла
            std::string fileHash = computeFileHash(filePath);

            if (isImageFile(filePath)) {
                // Является картинкой
                std::string cameraName = GetCameraName(filePath.string());

                if (cameraName != "NoExifData" && cameraName != "Error") {
                    // Имеет Exif данные
                    std::string sanitizedCameraName =
                        sanitizeFolderName(cameraName);
                    fs::path newDir = targetDirectory / sanitizedCameraName;
```

```
fs::create_directories(newDir);
```

8. Разработка алгоритма для сортировки фото на основе информации о дате создания.

Следующий код сортирует файлы из указанной директории по годам их создания, извлекая данные из EXIF-метаданных изображений, и организует их в соответствующие папки: для каждого года, для файлов без данных о годе (NoYearData) и для файлов, которые не являются изображениями (NotPhotos). Дубликаты определяются с помощью хеш-сумм, перемещаются в подпапки Duplicates, и для них создаются уникальные имена. Обработка выполняется параллельно в четырёх потоках для повышения производительности, с учётом ошибок, которые логируются в консоль.

Листинг 8. Добавление «SortByYear.cpp»

```
#include "SortByYear.h"
#include "../CommonFunctions/computeFileHash.h"
#include "../CommonFunctions/IsImageFile.h"
#include "../CommonFunctions/getUniquePath.h"
#include "../CommonFunctions/SanitizeFolderName.h"
#include "../GetPhotosParameters/GetYearOfPhoto.h"

void processFilesWithYear(
    const std::vector<fs::path>& files,
    const fs::path& targetDirectory,
    std::unordered_map<std::string, int>& hashMap
) {
    std::mutex coutMutex;
    static std::mutex hashMapMutex;

    for (const auto& filePath : files) {
        try {
            // Считаем хеш файла
            std::string fileHash = computeFileHash(filePath);

            if (isImageFile(filePath)) {
                // Является картинкой
                std::string year = GetYearOfPhoto(filePath.string());
```

На этом работа с .cpp файлами на C++ была завершена. Следом будут представлены наработки дизайна приложения на фреймворке Qt.

9. Функция main.cpp

Сперва был создан заголовочный файл, в котором хранилась функция для

запуска всего приложения, представленная в Листинге 9..

Листинг 9. Добавление «main.cpp»

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);

    // Создаём главное окно
    MainWindow mainWindow;
    mainWindow.show();

    return app.exec();
}
```

10. Создание первичного окна.

После чего было создано основное окно со всеми элементами: подбор папок для сортировки и места хранения, выбор способа сортировки. Также была добавлена проверка на заполненность всех окон (см. Листинг 10).

Листинг 10. Добавление «mainwindow.cpp»

```
#include "mainwindow.h"
#include "sortingwidget.h"
#include "loadingwidget.h"
#include <QPalette>
#include <QLinearGradient>
#include <QFileDialog>
#include <QDebug> // Для отладочного вывода
#include <QMessageBox> // Для сообщений об ошибках

MainWindow::MainWindow (QWidget *parent) : QWidget(parent),
sortingWidget(nullptr), loadingWidget(nullptr) {
    // Установка заголовка окна
    setWindowTitle("FileSend");
    setFixedSize(800, 500);

    // Настройка цвета и градиента фона
    QPalette palette;
    QLinearGradient gradient(0, 0, 0, 500); // Заменяем height() на
    фиксированную высоту
    gradient.setColorAt(0.0, QColor(255, 203, 243));
    gradient.setColorAt(1.0, QColor(128, 0, 128));
    palette.setBrush(QPalette::Window, QBrush(gradient));
    setAutoFillBackground(true);
    setPalette(palette);

    QVBoxLayout *mainLayout = new QVBoxLayout(this);
```



```

// Заголовок
QLabel *titleLabel = new QLabel("Выбор папки для сортировки", this);
titleLabel->setStyleSheet("font-size: 24px; color: rgb(128, 0, 128); font-
weight: bold;");
titleLabel->setAlignment(Qt::AlignCenter);

QLabel *subtitleLabel = new QLabel("Сортировка может занять некоторое
время...", this);
subtitleLabel->setStyleSheet("font-size: 16px; color: rgb(128, 0, 128);");
subtitleLabel->setAlignment(Qt::AlignCenter);
subtitleLabel->setContentsMargins(0, -30, 0, 0);

QLabel *pathLabel1 = new QLabel("Укажите путь к неотсортированной папке с
фото:", this);
pathLabel1->setStyleSheet("font-size: 16px; color: rgb(128, 0, 128); font-
weight: bold;");

```

11. Создание окна с загрузкой.

Затем было создано окно загрузки процесса сортировки файлов, для более комфортного и удобного ожидания завершения, код реализации которого представлен в Листинге 11.

Листинг 11. Добавление «loadingwidget.cpp»

```

#include "loadingwidget.h"
#include <QTimer>
#include <QMovie>
#include <QDesktopServices>

LoadingWidget::LoadingWidget(QWidget *parent) : QWidget(parent) {
    setWindowTitle("Загрузка...");
    setFixedSize(800, 500);

    // Настройка цвета и градиента фона
    QPalette palette;
    QLinearGradient gradient(0, 0, 0, 500); // Заменяем height() на
фиксированную высоту
    gradient.setColorAt(0.0, QColor(255, 203, 243));
    gradient.setColorAt(1.0, QColor(128, 0, 128));
    palette.setBrush(QPalette::Window, QBrush(gradient));
    setAutoFillBackground(true);
    setPalette(palette);
    // Создание Layout
    QVBoxLayout *layout = new QVBoxLayout(this);

    // Создание меток
    loadingLabel = new QLabel("Идёт сортировка фотографий...", this);
    loadingLabel->setStyleSheet("font-size: 24px; color: rgb(128, 0, 128);;
font-weight: bold;");

```

```
loadingLabel->setAlignment(Qt::AlignCenter);
layout->addWidget(loadingLabel);

// Инициализируем статус метку
statusLabel = new QLabel("Статус: Начало обработки", this);
statusLabel->setStyleSheet("font-size: 24px; color: rgb(128, 0, 128);; font-
weight: bold;");
statusLabel->setAlignment(Qt::AlignCenter);
layout->addWidget(statusLabel);

// Создаем вращающийся индикатор загрузки
spinnerLabel = new QLabel(this);
spinnerLabel->setAlignment(Qt::AlignCenter);
```

12. Переход к результату сортировки.

Следующим шагом было добавление кнопки «Перейти к результату», чтобы напрямую можно было перейти к папке в Проводнике, в которую сохранялись результаты сортировки (см. Листинг 12).

Листинг 12. Добавление кнопки в «loadingwidget.cpp»

5. Демонстрация работы приложения

После запуска приложение, перед пользователем открывается начальное окно, в котором он может провести все необходимые действия, чтобы сортировка была успешно проведена, а именно: указать путь к папке, в которую будут добавляться отсортированные файлы, выбрать с помощью радиокнопок метод сортировки и перейти далее («Начать сортировку») к сортировке (см. рис. 2).

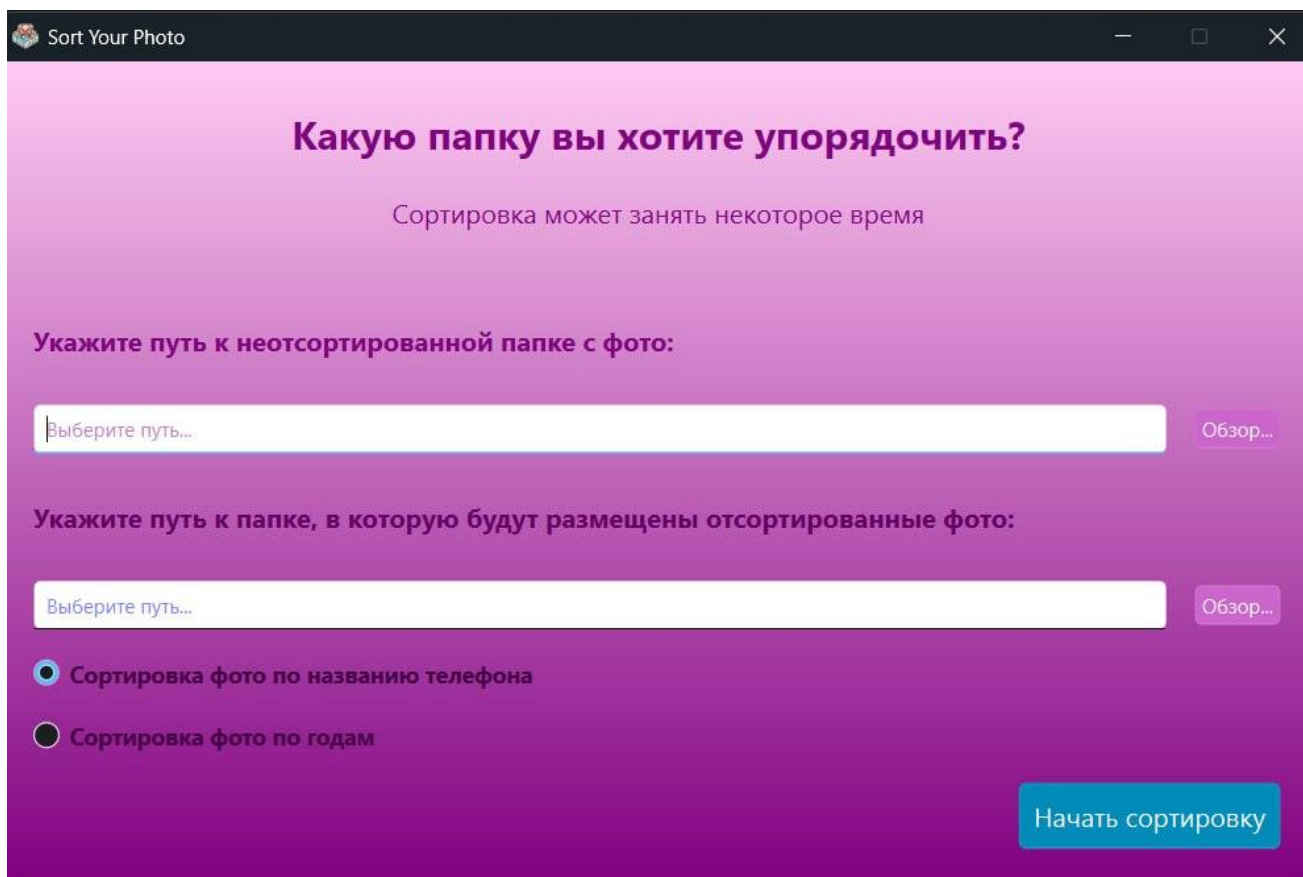


Рисунок 2 – Первая и основная страница

Одной из реализованных фич была проверка на заполненность всех полей и вывод сообщения об ошибке. Если пользователь не указал бы хоть что-то, то он бы не смог приступить к сортировке (см. рис. 3).

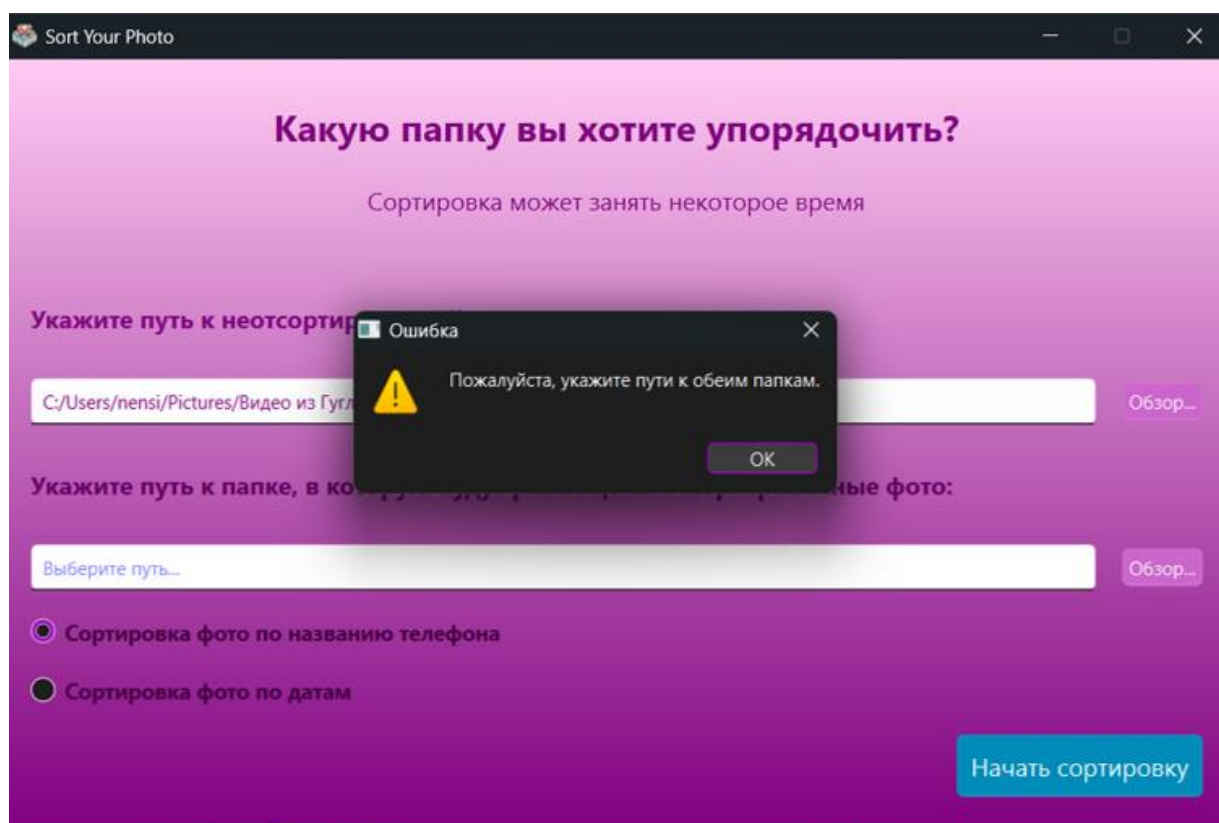


Рисунок 3 – Вывод ошибки из-за незаполненного поля

Далее будет выбрана папка для сохранения итогового результата сортировки (см. рис. 4).

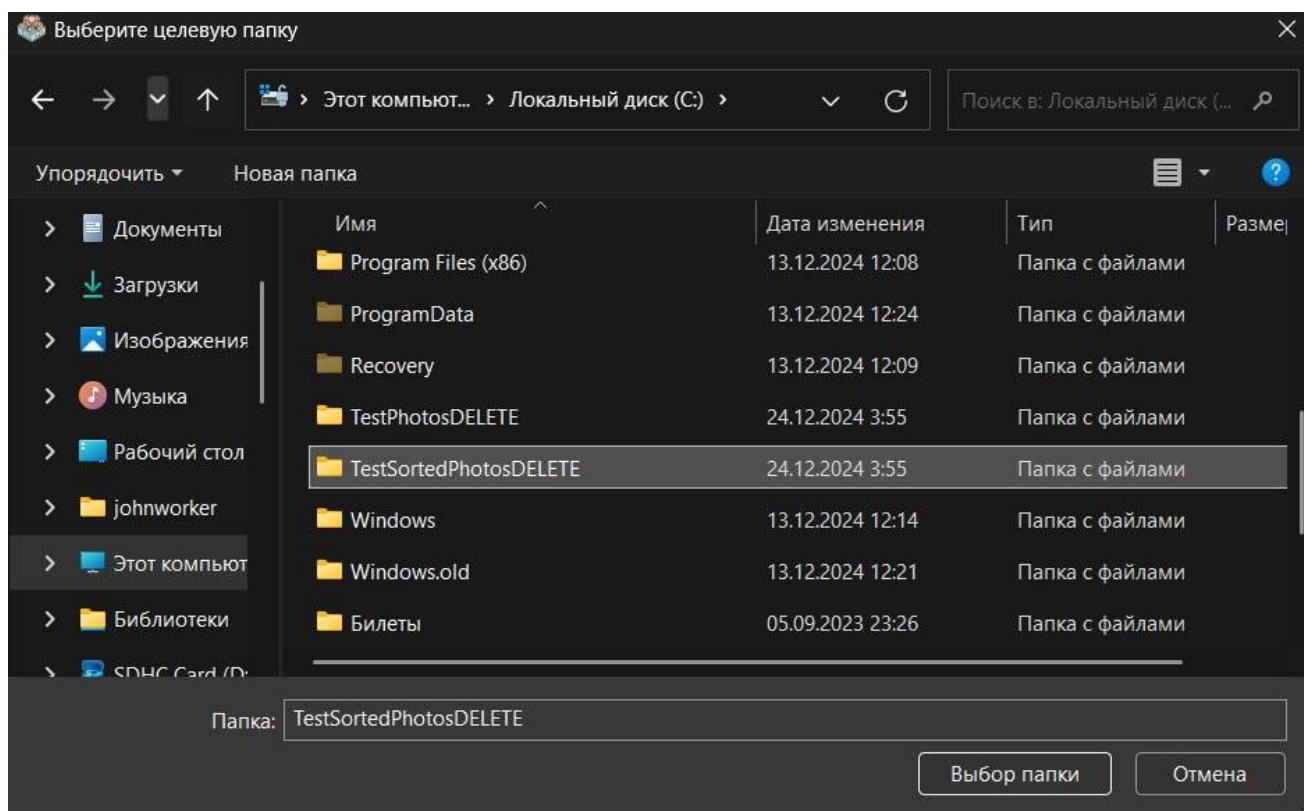


Рисунок 4 – Пример выбора папки в Проводнике

На рисунке 5 продемонстрировано, что все поля заполнены, метод сортировки выбран, значит можно переходить дальше.

Рисунок 5 – Заполнение полей путями к Проводнику

После успешного заполнения всех полей, был выполнен переход на окно с сортировкой, где прописывается, сколько прошло времени с начала загрузки. Также ожидание скрашивает индикатор загрузки – гифка (см. рис. 6).

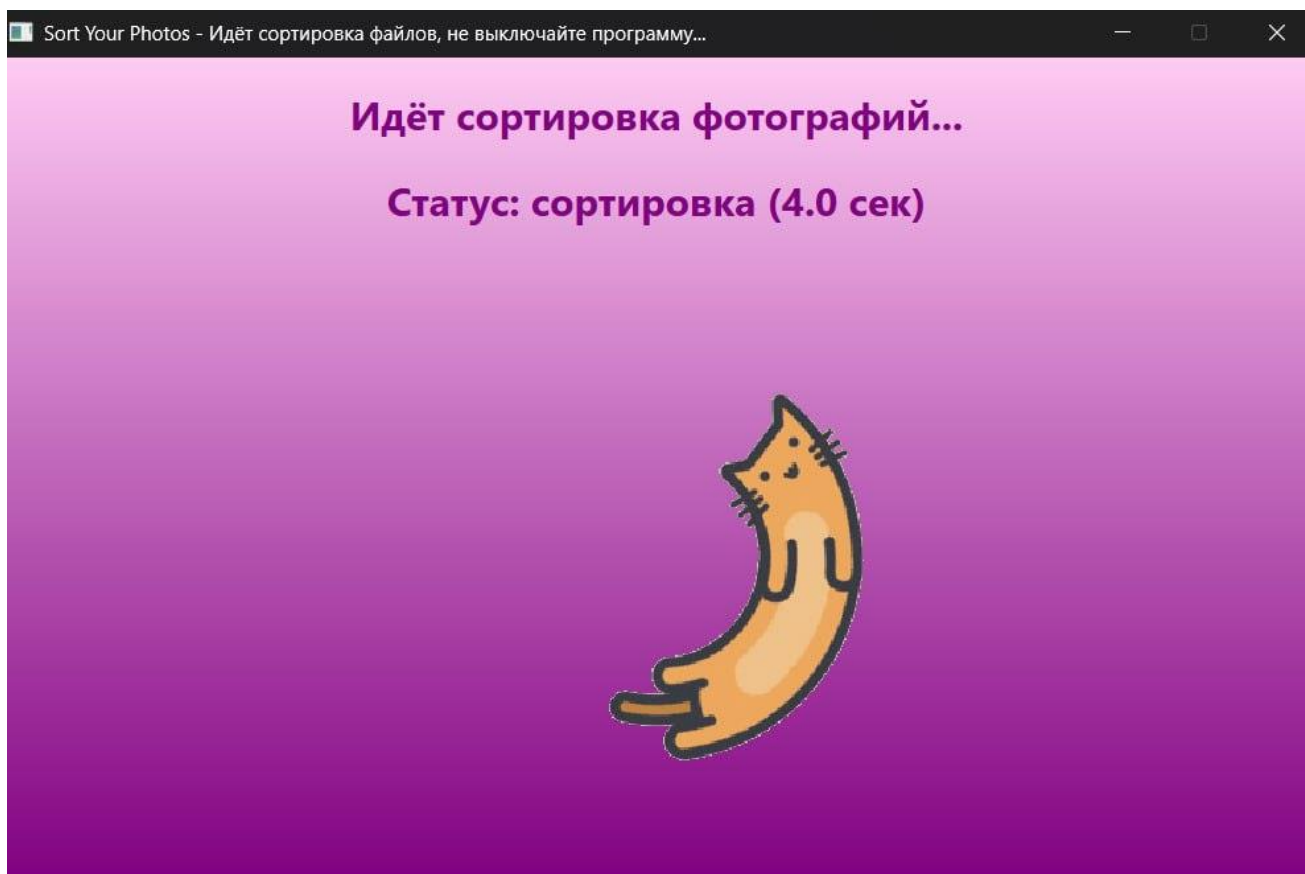


Рисунок 6 – Проведение сортировки файлов

После того, как приложение отсортирует все файлы, окно приложения сменится на сообщение о завершении загрузки и появится кнопка для перехода в проводник с отсортированными файлами (см. рис. 7).

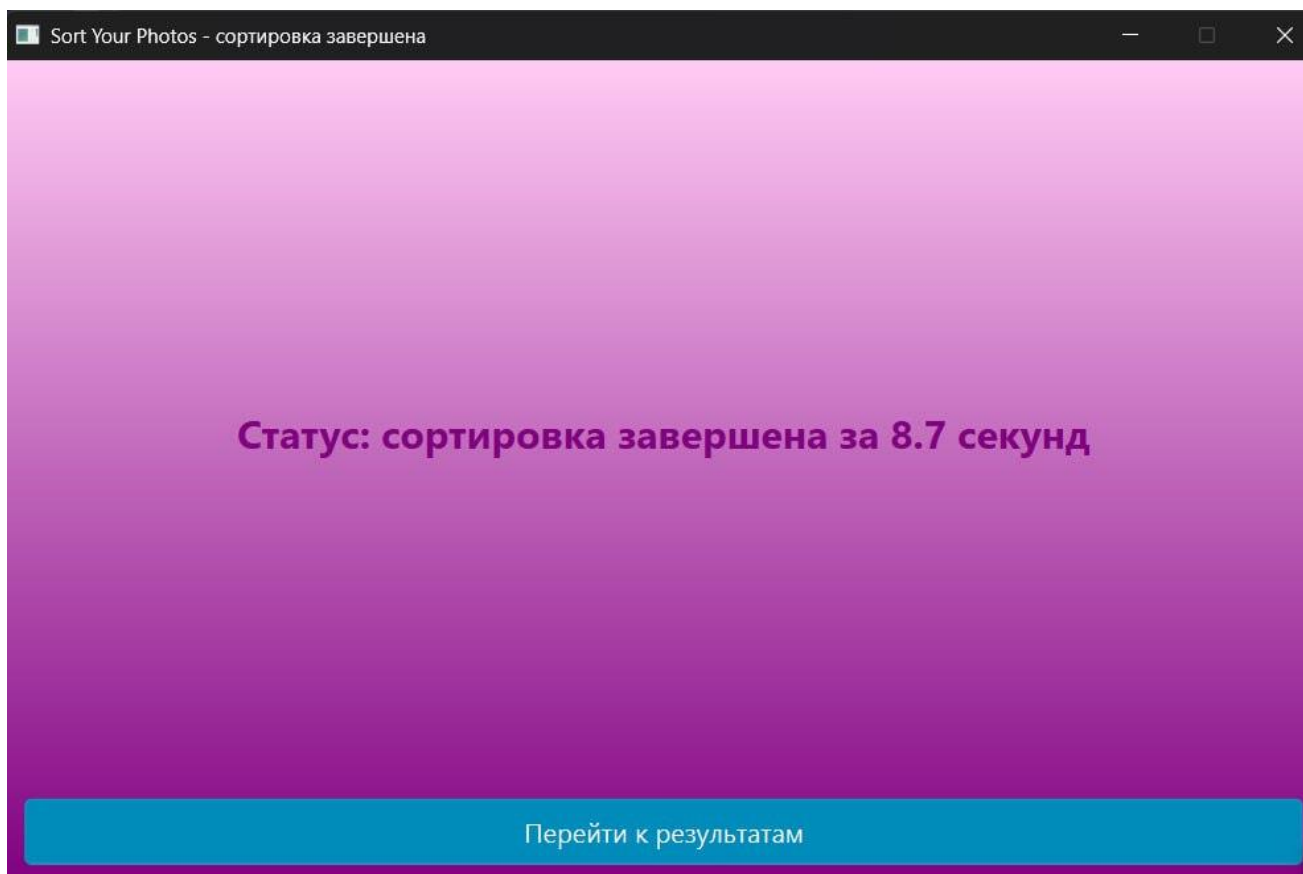


Рисунок 7 – Сортировка завершена

После перехода по кнопке, мы попадаем в Проводник, где можно увидеть отсортированные файлы (см. рис. 8).

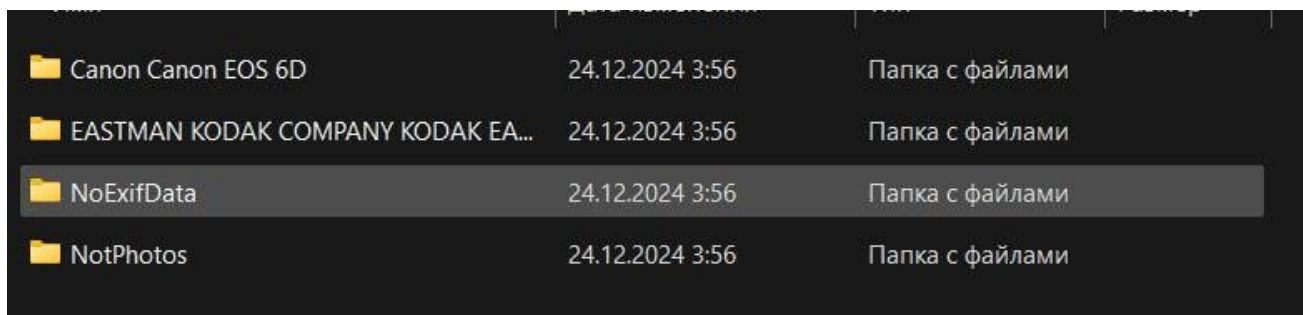


Рисунок 8 – Вывод результата сортировки в Проводнике

ЗАКЛЮЧЕНИЕ

Программа представляет собой удобный инструмент для упрощенной организации и сортировки фотографий в пользовательских папках. Простой интерфейс и ограниченные, но полезные функции делают ее полезной для широкого круга пользователей, позволяя настраивать порядок хранения фотографий и быстро находить нужные изображения.

В процессе работы реализованы алгоритмы сортировки фотографий по дате создания и устройству, на которое они были сделаны, а также механизмы определения дубликатов. Программа обладает широкими функциональными возможностями, включая распознавание форматов изображений, извлечение метаданных и обработку ошибок. Использование языка программирования C++ и фреймворка Qt обеспечило производительность и кроссплатформенность приложения.

В результате была создана удобная и надежная утилита, которая отвечает потребностям как профессиональных пользователей, так и обычных владельцев цифровых устройств, стремящихся упорядочить свои фотографии. Это приложение можно использовать как базу для дальнейшего развития, включая добавление новых функций и улучшение пользовательского опыта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная страница и документация библиотеки Exiv2. – Обновляется по мере выхода обновлений на библиотеку Exiv2. – URL: <https://exiv2.org/> (дата обращения: 10.10.2024).
2. Официальная страница и документация библиотеки OpenSSL. – Обновляется по мере выхода обновлений на библиотеку Exiv2. – URL: <https://openssl-library.org/> (дата обращения: 10.10.2024).
3. Официальная страница и документация IDE Microsoft Visual Studio. – Обновляется по мере выхода обновлений на IDE. – URL: <https://visualstudio.microsoft.com/ru/> (дата обращения: 01.10.2024).
4. Документация C++ и руководство работы с ним в Visual Studio. – Обновляется по мере выхода обновлений на C++. – URL: <https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170> (дата обращения: 01.10.2024).
5. Видеоурок «Подключение сторонних библиотек к проектам Visual Studio C++ 2022 #7». – Не обновляется. – URL: <https://www.youtube.com/watch?v=fybWjlNGxMQ> (дата обращения: 10.10.2024).
6. Бесплатный курс видеолекций "Основы C++. Программирование для начинающих". – Не обновляется. – URL: <https://www.youtube.com/playlist?list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r> (дата обращения: 01.10.2024).
7. Официальная страница и документация фреймворка Qt. – Обновляется по мере выхода обновлений на фреймворк Qt. – URL: <https://www.qt.io/offline-installers> (дата обращения: 10.10.2024).

ПРИЛОЖЕНИЕ А.

Техническое задание на разрабатываемое программное обеспечение.

1. Общие сведения

Наименование проекта:

Приложение по обработке и сортировке фото «Фото в порядке»

Основания для разработки:

Техническое задание разработано на основе курсового проекта и учебного задания по разработке программного обеспечения. Так же есть и практическое применение: необходимость упрощения управления большими коллекциями фотографий у пользователей смартфонов.

Разработчики:

Студенты группы 4217 – Д.М. Никитин, А.Н. Медянкина

Заказчик:

СПБГУАП, Кафедра 41, канд. техн. наук С.А. Чернышев.

2. Цель разработки

Разработка приложения для упрощения организации фотографий с возможностью сортировки по дате создания или названию устройства (телефона). Приложение должно быть интуитивно понятным и обеспечивать надежную обработку изображений, включая работу с метаданными.

3. Назначение разработки

Предоставить пользователям инструмент для удобной и быстрой сортировки фотографий на основе даты их создания или устройства, на которое они были сделаны. Это приложение направлено на решение задачи управления большими коллекциями изображений, которые накапливаются у пользователей, и упрощает их систематизацию.

4. Функции программного обеспечения

4.1 Сортировка по дате:

- Извлечение даты создания изображения из метаданных (EXIF).
- Перемещение фотографий в папки, организованные по годам их создания.

- Разделение фотографий без данных о дате в отдельную папку (NoYearData).

4.2 Сортировка по устройству:

- Извлечение информации об устройстве (производитель и модель камеры) из EXIF-метаданных.

- Разделение фотографий по папкам, названным в честь устройства, на которое сделан снимок.

- Перемещение фотографий без данных об устройстве в папку (NoExifData).

4.3 Работа с дубликатами:

- Автоматическое определение дубликатов на основе хэш-сумм файлов (SHA-256).

- Перемещение дубликатов в отдельные подпапки (Duplicates) внутри категорий.

4.4 Работа с не относящимися к фотографиям файлами

- Определение файлов, не являющихся изображениями (по расширению).

- Перемещение таких файлов в отдельную папку (NotPhotos).

4.5 Настройка путей для сортировки

- Выбор исходной папки с фотографиями.

- Выбор папки для сохранения отсортированных фотографий.

4.6 Поддержка различных форматов изображений

- Работа с популярными форматами изображений (JPEG, PNG, BMP и др.).

4.7 Уведомления о ходе процесса

- Индикация текущего статуса сортировки (начало, прогресс, завершение).

- Информация о времени, необходимом для выполнения сортировки.

4.8 Обработка ошибок

- Вывод сообщений о проблемах с доступом к файлам или отсутствием необходимых данных в EXIF-метаданных.
- Логирование ошибок для диагностики.

4.9 Интеграция с файловой системой

- Возможность открыть папку с результатами сортировки прямо из приложения.

4.10 Многопоточность для ускорения обработки

- Параллельная обработка файлов для повышения производительности при работе с большими архивами.

5. Требования к программному обеспечению

5.1 Функциональные требования:

- Программа должна корректно выполнять сортировку фотографий по дате создания или устройству, на которое был сделан снимок, используя EXIF-метаданные.
- Приложение должно поддерживать обработку дубликатов, файлов без метаданных и не относящихся к изображениям файлов, распределяя их в соответствующие папки.

5.2 Пользовательский интерфейс:

- Интерфейс приложения должен быть интуитивно понятным, с возможностью выбора папок и критериев сортировки.
- Приложение должно отображать прогресс выполнения задачи и предоставлять уведомления о завершении процесса сортировки.

5.3 Производительность:

- Программа должна поддерживать многопоточность (4 потока) для ускорения обработки больших объемов фотографий.
- Время обработки должно быть оптимизировано для каталогов, содержащих более 10 000 файлов.

6. Средства разработки и инструменты

6.1 Язык программирования:

- C++: основной язык для реализации функциональности и

обеспечения высокой производительности.

- Фреймворк для разработки интерфейса: Qt Framework: для создания графического пользовательского интерфейса (GUI), обработки событий и работы с файлами.

6.2 Библиотеки для работы с изображениями и метаданными:

- Exiv2: для извлечения EXIF-метаданных из фотографий (дата создания, информация о камере).

- OpenSSL: для генерации хэш-сумм файлов (SHA-256) для обработки дубликатов.

- Среда разработки (IDE):

- Qt Creator: для удобной работы с проектом, дизайна интерфейса и отладки.

- Visual Studio для написания и тестирования C++ кода.

6.3 Системы контроля версий:

- Git: для управления версионностью кода и совместной работы над проектом.

6.4 Инструменты для тестирования и отладки:

- Visual Studio: для отладки программы.

6.5 Платформа для сборки проекта:

- QMake и CMake: для упрощения конфигурации сборки проекта.

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

7.1 Техническая готовность:

Установка приложения на всех целевых устройствах и проверка совместимости оборудования и программного обеспечения с требованиями приложения.

7.2 Информационная подготовка:

Загрузка и корректировка исходных данных в системе, а так же настройка доступа и учетных записей для пользователей.

8. Требования к документированию

Для системы должны быть разработаны документация, описывающая структуру и функционал модулей.

9. Источники разработки

- Сайты Visual Studio, Exiv2, OpenSSL, Stakowerflow
- Open Source с наглядным использованием Qt на git hub
- Открытые источники, включая документацию Qt и примеры реализованных задач.

ПРИЛОЖЕНИЕ Б.

Методика приемо-сдаточных испытаний

1. Цель испытаний

Целью приемо-сдаточных испытаний является проверка функциональности, производительности, удобства использования и надежности приложения для сортировки фотографий. Испытания должны подтвердить, что приложение соответствует требованиям технического задания и готово к эксплуатации.

2. Общие положения

Приемо-сдаточные испытания включают в себя комплекс тестов, направленных на проверку всех ключевых функций приложения. Испытания проводятся в контролируемых условиях с использованием заранее подготовленных тестовых данных и сценариев. Результаты испытаний фиксируются и анализируются для принятия решения о готовности приложения к вводу в эксплуатацию.

3. Средства испытаний

- Тестовые данные: набор фотографий с различными метаданными (дата создания, информация о камере) и дубликатами.
- Тестовые сценарии: описание шагов и ожидаемых результатов для каждого теста.
- Системы контроля версий: Git для управления версиями кода и отслеживания изменений.

4. Порядок проведения испытаний

4.1 Предварительная подготовка

1. Создание набора фотографий с различными метаданными и дубликатами.
2. Установка библиотек и фреймворков на C++, настройка компилятора, размещение динамических библиотек в каталогах и подкаталогах проекта.

4.2 Этапы испытаний

4.2.1 Функциональное тестирование:

- Проверка корректности сортировки фотографий по дате создания.
- Проверка корректности сортировки фотографий по названию устройства.
- Проверка работы с дубликатами (определение и перемещение дубликатов в отдельные папки).
- Проверка обработки файлов без метаданных и не относящихся к фотографиям файлов.

4.2.2 Тестирование производительности:

- Измерение времени выполнения сортировки для различных объемов данных (3043 файла, 313 папок – 510 Мб, 7-8 сек).
- Проверка работы многопоточности и распределения нагрузки между потоками.

4.2.3 Тестирование удобства использования:

- Оценка интуитивности и удобства интерфейса приложения на примере использования приложения сторонним лицом (не разработчиком).
- Проверка корректности отображения прогресса выполнения сортировки и уведомлений о завершении процесса.

4.2.4 Тестирование надежности:

— Проверка работы приложения при различных сценариях ошибок (например, отсутствие доступа к файлам, поврежденные метаданные).

4.3 Оценка результатов

Анализ результатов тестирования: Сравнение фактических результатов с ожидаемыми, выявление отклонений и ошибок.

Фиксация результатов: Документирование результатов каждого теста, включая описание выявленных и рекомендации по их устранению.

Отчет о тестировании: Подготовка отчета, содержащего описание проведенных тестов, выявленные проблемы и общие выводы о готовности приложения к эксплуатации

5. Критерии приемки

Приложение считается готовым к эксплуатации, если:

— Все ключевые функции работают корректно и соответствуют требованиям технического задания: файлы сортируются корректно, дубликаты определяются, метаданные считываются корректно.

— Время выполнения сортировки соответствует заявленным показателям производительности.

— Интерфейс приложения интуитивно понятен и удобен для пользователя.

— Приложение корректно обрабатывает ошибки и предоставляет пользователю информацию о проблемах.

— Все выявленные в процессе тестирования проблемы устранены.

6. Оформление результатов

6.1 По итогам проведенных приемо-сдаточных испытаний подтверждено, что разработанное программное обеспечение полностью соответствует требованиям, изложенным в техническом задании. Протокол испытаний включает описание проведённых тестов, их результаты, выявленные дефекты, соответствие системы заявленным

требованиям и рекомендации по доработке.

6.2 Акт приемо-сдаточных испытаний: предоставляется отчет, подписываемый преподавателем.

ПРИЛОЖЕНИЕ В. Исходный код

Ссылка на репозиторий:

https://github.com/Diminasss/sort_your_photos/tree/main