**Руководство по установке и сборке To-Do Manager**

Это пошаговое руководство для тех, кто только что **склонировал** проект и хочет быстро установить зависимости, собрать и запустить приложение.

**1. Клонирование репозитория**

Откройте терминал (PowerShell на Windows или Bash на Linux/macOS) и выполните:

git clone https://github.com/yourusername/to\_do\_manager.git

cd to\_do\_manager

Проект склонируется в папку to\_do\_manager, и вы окажетесь в её корне.

**2. Установка зависимостей**

**Для Linux (Ubuntu/Debian)**

sudo apt update

sudo apt install -y build-essential cmake libsqlite3-dev doxygen clang-format

**Для macOS (Homebrew)**

brew update

brew install cmake sqlite doxygen clang-format

**Для Windows (PowerShell + vcpkg)**

1. **Установите vcpkg** (если ещё не установлен):
2. git clone https://github.com/microsoft/vcpkg.git C:\vcpkg
3. cd C:\vcpkg

.\bootstrap-vcpkg.bat

1. **Добавьте vcpkg в PATH** (опционально):

setx PATH "%PATH%;C:\vcpkg"

1. **Установите пакеты**:

vcpkg install nlohmann-json sqlite3 gtest doxygen clang-format

1. При вызове CMake укажите toolchain vcpkg (см. пункт 3 ниже).

**3. Сборка проекта через CMake**

Создаём каталог для сборки и запускаем CMake:

mkdir build

cd build

* **Linux/macOS**:

cmake ..

* **Windows** (с vcpkg):

cmake .. -DCMAKE\_TOOLCHAIN\_FILE=C:/vcpkg/scripts/buildsystems/vcpkg.cmake -DVCPKG\_TARGET\_TRIPLET=x64-windows

Затем выполняем сборку:

cmake --build . --config Release

Исполняемые файлы окажутся в:

* Linux/macOS: build/ToDoManager и build/ToDoTests
* Windows: build\Release\ToDoManager.exe и build\Release\ToDoTests.exe

**4. Прогон тестов**

В каталоге сборки выполните:

ctest -C Release --output-on-failure

Если ctest недоступен, запустите напрямую:

# Linux/macOS

./ToDoTests

# Windows (PowerShell)

cd Release

.

ToDoTests.exe --gtest\_color=yes

Все тесты должны пройти (0 неудачных).

**5. Генерация документации (Doxygen)**

Перейдите в папку docs и запустите:

cd docs

doxygen Doxyfile

* HTML-документация появится в docs/output/html/index.html.
* Откройте её в браузере для просмотра документации по классам и методам.

**6. Форматирование кода**

В корне проекта (где лежит файл .clang-format) выполните:

# Linux/macOS

clang-format -i -style=file src/\*.cpp src/\*.hpp tests/\*.cpp

# Windows (PowerShell)

clang-format -i -style=file src\\*.cpp src\\*.hpp tests\\*.cpp

После этого все файлы будут отформатированы по заданному стилю.

**7. Запуск приложения**

Перейдите в каталог с собранным бинарным:

# Linux/macOS

cd build

./ToDoManager add "Пример задачи" --due=2025-06-30 --tags=example,test

# Windows (PowerShell)

cd build\Release

.

ToDoManager.exe add "Пример задачи" --due=2025-06-30 --tags=example,test

**Основные команды CLI**:

* add <description> [--due YYYY-MM-DD] [--tags tag1,tag2] — добавить задачу
* remove <id> — удалить задачу
* done <id> — отметить выполненной
* list [--all|--done|--pending] — вывести задачи
* search <query> — поиск по описанию
* update-date <id> --due YYYY-MM-DD — обновить дедлайн
* export --format <json|csv> --out <path> — экспорт
* undo — отменить последнее
* глобальные опции: --data-file <path>, --store-format <json|sqlite>

НИЖЕ КОМАНДЫ ДЛЯ ДЕМОНСТРАЦИИ ПРЕПОДУ

# 7.1 Добавить

.\ToDoManager.exe add "Купить хлеб" --due=2025-06-10 --tags=food,home

# 7.2 Посмотреть все

.\ToDoManager.exe list --all

# 7.3 Отметить готовым

.\ToDoManager.exe done 1

# 7.4 Экспорт в CSV

.\ToDoManager.exe export --format csv --out tasks.csv

# 7.5 Undo

.\ToDoManager.exe undo

# 7.6 Удалить

.\ToDoManager.exe remove 1

# 7.7 Проверить лог

Get-Content history.log