Подлючаем необходимые библиотеки и + закидываем sqlite в исходные файлы проекта

------next slide-------

Русификация консоли, структуры бла бла бла,

Здесь происходит открытие базы данных SQLite в памяти. SQLite позволяет открывать базу данных не только из файла, но и в памяти компьютера. Для этого в качестве параметра в функцию **sqlite3\_open()** передается строка **":memory:"**, которая указывает, что база данных должна быть создана в памяти.

В данном случае результат открытия базы данных помещается в указатель **db**. Если открытие базы данных прошло успешно, то функция возвращает **SQLITE\_OK**, а если произошла ошибка, то она возвращается в **rc**. Если произошла ошибка при открытии базы данных, то выводится сообщение об ошибке и происходит закрытие базы данных и выход из программы с кодом возврата **rc**.

------next slide-------

Здесь создаются две таблицы в базе данных SQLite, если они еще не существуют. Первая таблица называется "data1" и содержит три столбца: "integer" типа INTEGER, "decimal" типа REAL и "text" типа TEXT. Вторая таблица называется "data2" и также содержит три столбца с теми же типами данных.

Для создания таблиц используется SQL-запрос "CREATE TABLE IF NOT EXISTS", который создает новую таблицу с указанными столбцами, если таблица еще не существует. Если возникнет ошибка при выполнении запроса, то программа выведет сообщение об ошибке и закроет базу данных.

------next slide-------

Здесь происходит добавление данных в таблицы **data1** и **data2**.

Первый шаг - создание строк SQL запросов для вставки данных в таблицы. Эти запросы формируются с помощью функции **std::to\_string**, которая преобразует числовые значения в строки, и конкатенации строк с помощью оператора **+**.

Второй шаг - выполнение SQL запросов с помощью функции **sqlite3\_exec**. Запросы передаются в качестве параметра **sql** в виде строки с помощью метода **c\_str()**.

Если запрос не был выполнен успешно (возвращает код **SQLITE\_OK**), выводится сообщение об ошибке в стандартный поток ошибок **std::cerr**. В этом случае база данных закрывается с помощью функции **sqlite3\_close**, а функция возвращает код ошибки. Если запросы были выполнены успешно, программа продолжит работу.

------next slide-------

Этот код выполняет запрос на умножение данных, хранящихся в двух таблицах "data1" и "data2" и затем выполняет проверку на успешность выполнения запроса.

Сначала создается запрос на умножение данных двух таблиц:

const char\* multiply\_query = "SELECT d1.integer \* d2.integer, d1.decimal \* d2.decimal, d1.text || d2.text FROM data1 AS d1, data2 AS d2;";

Затем создается объект sqlite3\_stmt\* stmt, который представляет собой подготовленное выражение SQL.

sqlite3\_stmt\* stmt = nullptr; rc = sqlite3\_prepare\_v2(db, multiply\_query, -1, &stmt, nullptr); if (rc != SQLITE\_OK) { std::cerr << "Can't prepare statement: " << sqlite3\_errmsg(db) << std::endl; sqlite3\_close(db); return rc; }

Функция sqlite3\_prepare\_v2() компилирует SQL-выражение в подготовленное выражение и возвращает его в переменной stmt.

Затем выполняется запрос:

rc = sqlite3\_step(stmt);

Функция sqlite3\_step() выполняет запрос, который был предварительно подготовлен функцией sqlite3\_prepare\_v2(). Если запрос выполнен успешно и возвращен хотя бы один ряд данных, то функция возвращает SQLITE\_ROW. Если запрос выполнен успешно, но возвращенных данных нет, то функция возвращает SQLITE\_DONE. Если произошла ошибка, то функция возвращает соответствующий код ошибки.

В этом коде проверяется, что запрос выполнен успешно и возвращен хотя бы один ряд данных:

if (rc != SQLITE\_ROW) { std::cerr << "Can't execute statement: " << sqlite3\_errmsg(db) << std::endl; sqlite3\_close(db); }

Если произошла ошибка, то выводится сообщение об ошибке и происходит закрытие базы данных.

------next slide-------

Этот код получает результаты запроса на умножение данных, который был выполнен в предыдущем коде.

В первых трех строках создается структура **Data**, которая заполняется значениями из столбцов, полученных из запроса при помощи функций **sqlite3\_column\_int**, **sqlite3\_column\_double** и **sqlite3\_column\_text**.

Затем значения **Data** выводятся в консоль при помощи функции **std::cout**. Выводятся значения **integer** и **decimal**, а также результат конкатенации строк из столбцов **text** таблиц **data1** и **data2**.

------next slide-------

Здесь закрывается соединение с базой данных и освобождаются занятые ей ресурсы. Функция **sqlite3\_finalize** используется для завершения подготовленного выражения, которое было создано с помощью **sqlite3\_prepare\_v2**. Затем функция **sqlite3\_close** закрывает соединение с базой данных, освобождая все связанные с ним ресурсы. Наконец, функция возвращает 0, чтобы показать, что программа завершилась без ошибок.