УО «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра «Информационных систем и технологий»

Специальность 1-40 05 01 «Информационные Системы и Технологии»

**Реферат**

**На тему «Разработка программы на C++ с использованием SQLite для обработки данных в структурах»**

|  |  |  |  |
| --- | --- | --- | --- |
| **Выполнил** |  |  |  |
| Студент 1 курса группы 1 |  |  | Д.И. Велютич |
|  | подпись, дата |  | инициалы и фамилия |
| **Проверил** |  |  | Н.И. Белодед |
|  | подпись, дата |  | инициалы и фамилия |

Минск 2023

Содержание

[Описание работы программы 2](#_Toc131559516)

[Программный код 4](#_Toc131559517)

[Скриншоты выполнения программы 6](#_Toc131559518)

В данном реферате рассматривается задача умножения элементов двух структур данных с использованием языка программирования C++. Приведен пример кода (см. главу “Исходный код”), в котором выполняется умножение чисел и конкатенация строк с использованием встроенной библиотеки SQLite для работы с базами данных.

1. Объявление структуры Data.

Структура Data представляет собой пользовательский тип данных, который содержит три поля: integer (целое число), decimal (дробное число) и text (строка). Эта структура используется для хранения данных, которые затем умножаются и объединяются.

1. Инициализация структур с данными.

В данном разделе инициализируются две структуры данных, data1 и data2, с предопределенными значениями. Эти значения будут использоваться для демонстрации работы программы.

1. Открытие и создание таблиц в базе данных.

Для работы с базами данных используется библиотека SQLite. В памяти создается временная база данных, в которой создаются две таблицы, data1 и data2, соответствующие структурам данных.

1. Добавление данных в таблицы.

Значения из структур данных data1 и data2 добавляются в соответствующие таблицы базы данных.

1. Выполнение запроса на умножение данных и конкатенацию строк.

С помощью SQL-запроса выполняется умножение значений целых и дробных чисел из двух таблиц, а также конкатенация строк. Результаты запроса сохраняются в новую структуру данных result.

1. Вывод результатов запроса.

На экран выводятся результаты умножения чисел и конкатенации строк, полученные в результате выполнения запроса.

1. Освобождение ресурсов.

После выполнения всех операций освобождаются ресурсы, связанные с базой данных и подготовленным запросом.

Данная программа на C++ демонстрирует умение манипулировать структурами данных, в частности числами и строками, с применением мощных функций языка программирования и взаимодействие с базой данных SQLite. При выполнении этой программы, вы сможете наблюдать умножение чисел и объединение строк из двух различных структур данных.

Это отражает гибкость и многофункциональность языка C++ при работе с разнообразными типами данных и структурами, а также использование встроенных библиотек для обработки и анализа информации, хранящейся в базах данных. Всё это сопровождается великолепным взаимодействием между различными компонентами, что обеспечивает эффективность и надежность в решении задач, связанных с обработкой данных.

# Программный код

#include <iostream>

#include <string>

#include <sqlite3.h> //SQLite

#include <iomanip> //Небольшое округление дробных чисел при большом количестве знаков после запятой

#include <Windows.h> //Русификация консоли

// Объявление структуры Data

struct Data {

int integer;

double decimal;

std::string text;

};

int main() {

// Установка кодировки консоли

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

// Инициализация структур с данными

const Data data1 = { 11, 1.8, "строка1" }; //вводить сюда целые числа, дробные числа, строки соответсвенно

const Data data2 = { 7, 3.14, "строка2" }; //сюда тоже

// Открываем базу данных в памяти

sqlite3\* db = nullptr;

int rc = sqlite3\_open(":memory:", &db);

if (rc != SQLITE\_OK) {

std::cerr << "Can't open database: " << sqlite3\_errmsg(db) << std::endl;

sqlite3\_close(db);

return rc;

}

// Создаем таблицы

const char\* create\_table = "CREATE TABLE IF NOT EXISTS data1 (integer INTEGER, decimal REAL, text TEXT); "

"CREATE TABLE IF NOT EXISTS data2 (integer INTEGER, decimal REAL, text TEXT);";

rc = sqlite3\_exec(db, create\_table, nullptr, nullptr, nullptr);

if (rc != SQLITE\_OK) {

std::cerr << "Can't create table: " << sqlite3\_errmsg(db) << std::endl;

sqlite3\_close(db);

return rc;

}

// Добавляем данные в таблицы

std::string insert\_data1 = "INSERT INTO data1 (integer, decimal, text) VALUES (" + std::to\_string(data1.integer) + ", " + std::to\_string(data1.decimal) + ", '" + data1.text + "');";

std::string insert\_data2 = "INSERT INTO data2 (integer, decimal, text) VALUES (" + std::to\_string(data2.integer) + ", " + std::to\_string(data2.decimal) + ", '" + data2.text + "');";

rc = sqlite3\_exec(db, insert\_data1.c\_str(), nullptr, nullptr, nullptr);

if (rc != SQLITE\_OK) {

std::cerr << "Can't insert data into table: " << sqlite3\_errmsg(db) << std::endl;

sqlite3\_close(db);

return rc;

}

rc = sqlite3\_exec(db, insert\_data2.c\_str(), nullptr, nullptr, nullptr);

if (rc != SQLITE\_OK) {

std::cerr << "Can't insert data into table: " << sqlite3\_errmsg(db) << std::endl;

sqlite3\_close(db);

return rc;

}

// Выполняем запрос на умножение данных

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

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;

}

rc = sqlite3\_step(stmt);

if (rc != SQLITE\_ROW) {

std::cerr << "Can't execute statement: " << sqlite3\_errmsg(db) << std::endl;

sqlite3\_close(db);

}

// Получаем результаты запроса

const Data result = { int(sqlite3\_column\_int(stmt, 0)),

double(sqlite3\_column\_double(stmt, 1)),

std::string(reinterpret\_cast<const char\*>(sqlite3\_column\_text(stmt, 2)))

};

// Выводим результаты запроса

std::cout << "Результат (3-я структура): " << std::endl;

std::cout << "Целые числа: " << result.integer << std::endl;

std::cout << "Дробные числа: " << result.decimal << std::endl;

std::cout << "Текст (конкатенация строк): " << result.text << std::endl;

// Освобождаем ресурсы

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

Ознакомиться с полной версией программы вы можете по [ссылке](https://disk.yandex.com/d/41S93FOlOlAWDw), либо скачав архив в письме!

# Скриншоты выполнения программы

1) Заполняем data1 и data2 нужными нам данными:



2) Запускаем программу и видим результат:

