Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

РАЗРАБОТКА ПРИЛОЖЕНИЯ КАЛЬКУЛЯТОР ФИЕСТЫ ДЕЛЮКС

Пояснительная записка к лабораторной работе по учебной дисциплине «Основы программирования: Алгоритмические языки и программирование»

по специальности 09.03.01 Информатика и вычислительная техника  
ПТИ.ЛР 4093 004.021ПЗ

|  |  |
| --- | --- |
|  | Руководитель  / И. Ю. Кулаков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  | Студент группы 4093  / М. Х. Шокиров  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

Содержание

[Содержание 2](#_Toc197987823)

[1 Постановка задачи 3](#_Toc197987824)

[1.1 Наименование задачи 3](#_Toc197987825)

[1.2 Словесное описание 3](#_Toc197987826)

[1.3 Внешние спецификации функции данных 3](#_Toc197987827)

[1.4 Специфика интерфейса 3](#_Toc197987828)

[1.5 Внешние данные тестирования 4](#_Toc197987829)

[1.6 Пример работы программы 4](#_Toc197987830)

[2 Проектирование задачи 5](#_Toc197987831)

[2.1 Наименование программы 5](#_Toc197987832)

[2.2 Уточнение словесного описания 5](#_Toc197987833)

[2.3 Декомпозиция функции 5](#_Toc197987834)

[Заключение 6](#_Toc197987835)

[Приложение А 7](#_Toc197987836)

[Приложение Б 15](#_Toc197987837)

[Приложение В 16](#_Toc197987838)

[Приложение Г 21](#_Toc197987839)

# 1 Постановка задачи

# Наименование задачи

Программа *«*Калькулятор Фиесты Делюкс*»* предназначена для выполнения арифметических операций с ингредиентами (сложение, вычитание, умножение, деление), проверки равенства длин последовательностей и сохранения данных в SQLite-базе.

# 1.2 Словесное описание

Цель работы — автоматизация расчетов для организации фиесты:

– выполнение операций с векторами ингредиентов;

– сохранение рецептов и запасов в базу данных;

– экспорт данных в CSV;

– поддержка типов int, float, double.

# 1.3 Внешние спецификации функции данных

Арифметические операции:

– сложение и вычитание;

– умножение и деление на число;

– проверка равенства длин.

Работа с данными:

– сохранение ингредиентов в SQLite;

– экспорт в CSV.

# 1.4 Специфика интерфейса

Программа разработана с графическим интерфейсом. Пользователь может взаимодействовать с программой через кнопки, поля ввода и визуальные элементы управления. Реализованы проверки корректности введённых данных и отображение соответствующих сообщений об ошибках.

# 1.5 Внешние данные тестирования

Для обеспечения надежности и корректной работы приложения были разработаны тестовые сценарии, охватывающие различные аспекты взаимодействия пользователя с системой:

Учтены возможные ошибки при вводе данных:

– некорректные форматы данных;

– граничные значения;

– ввод пустых данных.

# 1.6 Пример работы программы

Таблицы тест кейсов представлены в приложении А.2-А.5.

Таблицы Unit-тестирований представлены в приложении А.6.

# 2 Проектирование задачи

# 2.1 Наименование программы

Название программы: "Калькулятор Фиесты Делюкс".

# 2.2 Уточнение словесного описания

Программа «Калькулятор Фиесты Делюкс» представляет собой специализированный инструмент для кулинарных расчетов, разработанный для Кота в сапогах. Приложение состоит из трех ключевых модулей:

Модуль вычислений:

– реализует арифметические операции с векторами ингредиентов (сложение, вычитание, умножение, деление);

– поддерживает типы данных int, float, double;

– включает проверку ошибок (деление на ноль, неравные длины векторов).

Модуль работы с данными:

– сохраняет ингредиенты и рецепты в SQLite-базу;

– обеспечивает экспорт данных в CSV-формат для внешнего использования;

– автоматически создает необходимые файлы.

Пользовательский интерфейс:

– поля ввода: название ингредиента, количество;

– кнопки операций: Calcular, Guardar, Exportar a CSV;

– область вывода результатов в виде текста.

Программа разработана на C++ с использованием библиотек GTKmm и SQLite3, что обеспечивает кроссплатформенность и надежность.

# 2.3 Декомпозиция функции

Таблица функций программы представлена в приложении А.1.

# Заключение

В ходе выполнения работы было разработано приложение «Калькулятор Фиесты Делюкс», полностью соответствующее техническому заданию. Программа предоставляет удобный инструмент для выполнения кулинарных расчетов, включая базовые арифметические операции с ингредиентами, проверку данных и сохранение результатов. Приложение демонстрирует эффективное применение шаблонов C++, работу с базами данных и современными библиотеками для создания GUI. Проект может быть расширен за счет добавления новых операций, интеграции с другими форматами данных или улучшения визуализации результатов.

# Приложение А

(Обязательное)

Таблица А.1 – Основные объекты программы, запрашиваемые у пользователя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| Поле ввода ингредиента | entry\_ingredient | Gtk::Entry | Текст (до 100 символов) | Простая | Вход |
| Поле ввода количества | entry\_quantity | Gtk::Entry | int, float, double > 0 | Простая | Вход |
| Выбор операции | combo\_operation | Gtk::ComboBoxText | Варианты: +, -, \*, /, == | Простая | Вход |
| Кнопка вычисления | btn\_calculate | Gtk::Button | –– | Простая | Вход |
| Кнопка сохранения | btn\_save | Gtk::Button | –– | Простая | Вход |
| Кнопка экспорта | btn\_export | Gtk::Button | –– | Простая | Вход |
| Таблица результатов | text\_results | Gtk::TextView | Текст | Структура | Выход |
| База данных | db | Database | Файл data/fiesta.db | Структура | Выход |
| CSV-файл | exports/ingredients.csv | std::ofstream | Формат: name,quantity | Структура | Выход |

Таблица А.2 – Тест кейс для проверки арифметических операций программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID/Priority | | TC001 | | | 1 |
| IDEA: Проверка корректности арифметических операций | | | | | |
| Revision History | | | | | |
| Created on 12/05/2025 by Шокиров Мубинджон | | | Новый тест кейс | | |
| Precondition | | | | | |
| Программа запущена | | | | | |
| Список команд | Ожидаемый результат | | | Результат теста | |
| Ввести: 1.0, 2.0, выбрать операцию «Умножить», ввести множитель: 3 | Результат: 3.0, 6.0 | | | Passed | |
| Ввести: 5.0, 10.0, выбрать операцию «Делить», ввести делитель: 0 | Ошибка: «Error! División por cero» | | | Passed | |
| Ввести: 1, 2 и 3, 4, выбрать «Сложить» | Результат: 1, 2, 3, 4 | | | Passed | |

Таблица А.3 – Тест кейс для проверки работы с базой данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID/Priority | | TC002 | | | 2 |
| IDEA: Проверка сохранения и загрузки ингредиентов | | | | | |
| Revision History | | | | | |
| Created on 12/05/2025 by Шокиров Мубинджон | | | Новый тест кейс | | |
| Precondition | | | | | |
| Программа запущена, база данных fiesta.db пуста | | | | | |
| Список команд | Ожидаемый результат | | | Результат теста | |
| Ввести ингредиент: Сальса, количество: 100, нажать «Guardar» | В БД появляется запись: Сальса, 100.0 | | | Passed | |
| Повторно ввести Сальса, количество: 50 | В БД две записи Сальса | | | Passed | |
| Нажать «Exportar a CSV» | Файл ingredients.csv создан | | | Passed | |

Таблица А.4 – Тест кейс для тестирования валидации ввода

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID/Priority | | TC003 | | | 3 |
| IDEA: Проверка обработки некорректных данных | | | | | |
| Revision History | | | | | |
| Created on 12/05/2025 by Шокиров Мубинджон | | | Новый тест кейс | | |
| Precondition | | | | | |
| Программа AddressBook.exe запущена, открыто окно ввода ингредиентов | | | | | |
| Список команд | Ожидаемый результат | | | Результат теста | |
| Оставить поле «Cantidad» пустым, нажать «Guardar» | Ошибка: «Поле обязательно» | | | Passed | |
| Ввести «Cantidad»: abc, нажать «Calcular» | Ошибка: «Не число» | | | Passed | |
| Ввести: 1, 2, abc, выбрать «Умножить» | Ошибка: «Некорректный ввод» | | | Failed | |

Таблица А.5 – Тест кейс для проверки экспорта в CSV

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID/Priority | | TC004 | | | 4 |
| IDEA: Проверка создания CSV-файла | | | | | |
| Revision History | | | | | |
| Created on 12/05/2025 by Шокиров Мубинджон | | | Новый тест кейс | | |
| Precondition | | | | | |
| Программа AddressBook.exe запущена, в БД есть 3 ингредиента | | | | | |
| Список команд | Ожидаемый результат | | | Результат теста | |
| Нажать «Exportar a CSV» | Файл exports/ingredients.csv содержит 3 строки | | | Passed | |
| Указать несуществующую папку: C:/temp/missing\_folder/out.cs | Ошибка: «Не удалось создать файл» | | | Passed | |
| Проверить содержимое CSV | Столбцы: name,quantity; данные совпадают с БД | | | Passed | |

Таблица A.6 – Unit-тестирование рекурсивной функции

|  |  |  |  |
| --- | --- | --- | --- |
| Unit-Test ID | Вводные данные | Ожидаемый результат | Фактический результат |
| Базовый случай | | | |
| UT1 | add(1.0, 2.0, 3.0, 4.0) | 4.0, 6.0 | 4.0, 6.0 |
| UT2 | subtract(5.0, 3.0, 1.0, 2.0) | 4.0, 1.0 | 4.0, 1.0 |
| UT3 | multiply(1.5, 2.5, 2) | 3.0, 5.0 | 3.0, 5.0 |
| Проверка ошибок | | | |
| UT4 | divide(10.0, 20.0, 0) | Исключение: División por cero | Исключение получено |
| UT5 | areLengthsEqual(1, 2, 1, 2, 3) | Ошибка | Ошибка |
| Граничные случаи | | | |
| UT6 | multiply({}, 5) | пустой вектор | пустой вектор |
| UT7 | add(1e10, 2e10, 1e10, 2e10) | переполнение | переполнение |

# Приложение Б

(Обязательное)

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок Б.1 — Тестирование проекта

# Приложение В

(Обязательное)

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.1 — Блок-схема добавления контакта

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.2 — Блок-схема поиска контактов

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.3 — Блок схема редактирование контакта

Изображение выглядит как текст, снимок экрана, Шрифт, визитная карточка

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.4 — Блок схема удаление контакта

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок В.5 — Блок-схема экспорта контактов в CSV

# Приложение Г

(Обязательное)

Листинг Г.1 — код файла CMakeLists.txt:

cmake\_minimum\_required(VERSION 3.10)

project(FiestaCalculator)

set(CMAKE\_CXX\_STANDARD 17)

find\_package(PkgConfig REQUIRED)

pkg\_check\_modules(GTKMM REQUIRED gtkmm-3.0)

include(FetchContent)

FetchContent\_Declare(

googletest

URL https://github.com/google/googletest/archive/refs/heads/main.zip

)

FetchContent\_MakeAvailable(googletest)

file(MAKE\_DIRECTORY ${CMAKE\_SOURCE\_DIR}/data)

file(MAKE\_DIRECTORY ${CMAKE\_SOURCE\_DIR}/exports)

add\_executable(fiesta\_calculator

src/main.cpp

src/GUI.h

src/Calculator.h

src/Database.h

)

add\_executable(tests

tests/CalculatorTest.cpp

tests/DatabaseTest.cpp

src/Calculator.h

src/Database.h

)

target\_include\_directories(fiesta\_calculator PRIVATE ${GTKMM\_INCLUDE\_DIRS})

target\_link\_libraries(fiesta\_calculator ${GTKMM\_LIBRARIES} sqlite3)

target\_include\_directories(tests PRIVATE

${CMAKE\_SOURCE\_DIR}/src

)

target\_link\_libraries(tests

GTest::gtest\_main

sqlite3

)

enable\_testing()

add\_test(NAME AllTests COMMAND tests)

Листинг Г.2 — код файла Calculator.h:

#pragma once

#include <vector>

#include <stdexcept>

#include <algorithm>

#include <iostream>

template <typename T>

class FiestaCalculator {

public:

// Сложение (объединение запасов)

std::vector<T> add(const std::vector<T>& a, const std::vector<T>& b) const {

std::vector<T> result = a;

result.insert(result.end(), b.begin(), b.end());

return result;

}

// Вычитание (испорченные продукты)

std::vector<T> subtract(const std::vector<T>& a, const std::vector<T>& b) const {

if (a.size() != b.size()) {

throw std::runtime\_error("?Error! Разные размеры последовательностей");

}

std::vector<T> result;

for (size\_t i = 0; i < a.size(); ++i) {

result.push\_back(a[i] - b[i]);

}

return result;

}

// Умножение (масштабирование рецепта)

std::vector<T> multiply(const std::vector<T>& vec, T factor) const {

std::vector<T> result;

for (const auto& item : vec) {

result.push\_back(item \* factor);

}

return result;

}

// Деление (поровну между гостями)

std::vector<T> divide(const std::vector<T>& vec, T divisor) const {

checkDivision(divisor);

return multiply(vec, 1 / divisor);

}

// Проверка равенства длин

bool areLengthsEqual(const std::vector<T>& a, const std::vector<T>& b) const {

return a.size() == b.size();

}

protected:

void checkDivision(T divisor) const {

if (divisor == 0) {

throw std::runtime\_error("?Error! Divisi?n por cero");

}

}

};

Листинг Г.3 — код файла Database.h:

#pragma once

#include <sqlite3.h>

#include <string>

#include <vector>

#include <filesystem>

#include <fstream>

#include <iostream>

class Database {

public:

Database(const std::string& db\_path) {

std::filesystem::create\_directory("data");

if (sqlite3\_open(db\_path.c\_str(), &db) != SQLITE\_OK) {

throw std::runtime\_error("Error opening database: " + std::string(sqlite3\_errmsg(db)));

}

createTables();

}

~Database() { sqlite3\_close(db); }

void saveIngredient(const std::string& name, double quantity) {

std::string sql = "INSERT INTO Ingredients (name, quantity) VALUES (?, ?);";

sqlite3\_stmt\* stmt;

sqlite3\_prepare\_v2(db, sql.c\_str(), -1, &stmt, nullptr);

sqlite3\_bind\_text(stmt, 1, name.c\_str(), -1, SQLITE\_TRANSIENT);

sqlite3\_bind\_double(stmt, 2, quantity);

sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

}

std::vector<std::pair<std::string, double>> getIngredients() {

std::vector<std::pair<std::string, double>> ingredients;

sqlite3\_stmt\* stmt;

const char\* sql = "SELECT name, quantity FROM Ingredients;";

sqlite3\_prepare\_v2(db, sql, -1, &stmt, nullptr);

while (sqlite3\_step(stmt) == SQLITE\_ROW) {

ingredients.emplace\_back(

reinterpret\_cast<const char\*>(sqlite3\_column\_text(stmt, 0)),

sqlite3\_column\_double(stmt, 1)

);

}

sqlite3\_finalize(stmt);

return ingredients;

}

void exportToCSV(const std::string& filename) {

std::filesystem::create\_directory("exports");

std::ofstream file(filename);

if (!file.is\_open()) {

throw std::runtime\_error("Не удалось открыть файл для записи: " + filename);

}

file << "name,quantity\n";

auto ingredients = getIngredients();

for (const auto& [name, qty] : ingredients) {

file << name << "," << qty << "\n";

}

file.close();

std::cout << "Файл успешно создан: " << filename << std::endl;

}

private:

sqlite3\* db;

void createTables() {

const char\* sql =

"CREATE TABLE IF NOT EXISTS Ingredients ("

"id INTEGER PRIMARY KEY AUTOINCREMENT,"

"name TEXT NOT NULL,"

"quantity REAL NOT NULL);";

sqlite3\_exec(db, sql, nullptr, nullptr, nullptr);

}

};

Листинг Г.4 — код файла GUI.h:

#pragma once

#include <gtkmm.h>

#include "Calculator.h"

#include "Database.h"

#include <iostream>

class FiestaWindow : public Gtk::Window {

public:

FiestaWindow()

: db("data/fiesta.db"),

main\_box(Gtk::ORIENTATION\_VERTICAL, 10),

btn\_calculate("Calcular"),

btn\_save("Guardar"),

btn\_export("Exportar a CSV") {

set\_title("Калькулятор Фиесты Делюкс");

set\_default\_size(500, 400);

setupInputs();

setupButtons();

setupResults();

add(main\_box);

show\_all\_children();

}

private:

Database db;

FiestaCalculator<double> calculator;

Gtk::Box main\_box;

Gtk::Entry entry\_ingredient, entry\_quantity;

Gtk::ComboBoxText combo\_operation;

Gtk::Button btn\_calculate, btn\_save, btn\_export;

Gtk::Label label\_result;

Gtk::ScrolledWindow scroll\_results;

Gtk::TextView text\_results;

void setupInputs() {

entry\_ingredient.set\_placeholder\_text("Ingrediente");

entry\_quantity.set\_placeholder\_text("Cantidad");

combo\_operation.append("Sumar");

combo\_operation.append("Restar");

combo\_operation.append("Multiplicar");

combo\_operation.append("Dividir");

combo\_operation.set\_active(0);

main\_box.pack\_start(entry\_ingredient, Gtk::PACK\_SHRINK);

main\_box.pack\_start(entry\_quantity, Gtk::PACK\_SHRINK);

main\_box.pack\_start(combo\_operation, Gtk::PACK\_SHRINK);

}

void setupButtons() {

btn\_calculate.signal\_clicked().connect([this]() {

try {

double value = std::stod(entry\_quantity.get\_text());

std::vector<double> vec = { 1.0, 2.0, 3.0 };

std::string op = combo\_operation.get\_active\_text();

std::vector<double> result;

if (op == "Sumar") result = calculator.add(vec, { value, value, value });

else if (op == "Restar") result = calculator.subtract(vec, { value, value, value });

else if (op == "Multiplicar") result = calculator.multiply(vec, value);

else if (op == "Dividir") result = calculator.divide(vec, value);

text\_results.get\_buffer()->set\_text("Resultado: " + vectorToString(result));

}

catch (const std::exception& e) {

text\_results.get\_buffer()->set\_text("?Error! " + std::string(e.what()));

}

});

btn\_save.signal\_clicked().connect([this]() {

try {

db.saveIngredient(

entry\_ingredient.get\_text(),

std::stod(entry\_quantity.get\_text())

);

updateResults();

}

catch (...) {

text\_results.get\_buffer()->set\_text("Error al guardar");

}

auto ingredients = db.getIngredients();

if (ingredients.empty()) {

std::cout << "БД пуста! Нет данных для экспорта." << std::endl;

}

});

btn\_export.signal\_clicked().connect([this]() {

std::string csv\_path = "exports/ingredients.csv";

std::cout << "Путь экспорта: " << std::filesystem::absolute(csv\_path) << std::endl;

db.exportToCSV(csv\_path);

text\_results.get\_buffer()->set\_text("Datos exportados a: " + csv\_path);

});

main\_box.pack\_start(btn\_calculate, Gtk::PACK\_SHRINK);

main\_box.pack\_start(btn\_save, Gtk::PACK\_SHRINK);

main\_box.pack\_start(btn\_export, Gtk::PACK\_SHRINK);

}

void setupResults() {

text\_results.set\_editable(false);

scroll\_results.add(text\_results);

scroll\_results.set\_policy(Gtk::POLICY\_AUTOMATIC, Gtk::POLICY\_AUTOMATIC);

main\_box.pack\_start(scroll\_results);

}

void updateResults() {

auto ingredients = db.getIngredients();

std::string text;

for (const auto& ingredient : ingredients) {

text += ingredient.first + ": " + std::to\_string(ingredient.second) + "\n";

}

text\_results.get\_buffer()->set\_text(text.empty() ? "No hay datos" : text);

}

std::string vectorToString(const std::vector<double>& vec) {

std::string result;

for (double v : vec) {

result += std::to\_string(v) + " ";

}

return result;

}

};

Листинг Г.5 — код файла main.cpp:

#include "GUI.h"

#include <iostream>

int main(int argc, char\* argv[]) {

    std::cout << "База данных: " << std::filesystem::absolute("data/fiesta.db") << std::endl;

    auto app = Gtk::Application::create(argc, argv, "org.gtkmm.fiesta");

    FiestaWindow window;

    return app->run(window);

}