Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

Пояснительная записка
к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом нисходящего программирования. Программа для модульного тестирования»

Выполнил:
студенты группы 25ВВВ4
Егоров Е.С.

Принял.
к.т.н. Токарев А.Н.

13.05.2026

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

_____ М.А. Митрохин

«____»_____20__г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Егорову Егору Сергеевичу Группа 25ВВВ4

Тема проекта: ***Разработка программы сложной структуры методом нисходящего программирования.***

Исходные данные (технические требования) на проектирование

Программа для модульного тестирования
Требования: добавление записей, удаление тестов, редактирование тестов, поиск и сортировка тестов, фильтрация тестов по определённому критерию, запуск существующих тестов, отражение результатов прохождения тестов

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. **Расчетная часть**

Разработка программы.

2. **Графическая часть**

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. **Экспериментальная часть**

Отладка программы

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "19" 02.2026

Дата защиты проекта "____" _____

Руководитель _____

Задание получил "25" февраля 2026 г.

Студент Егоров Е. _____

Содержание

Введение..... 5
Постановка задачи 6
Выбор решения..... 8
Описание разработки программы 11
Описание программы..... 14
Руководство пользователя..... 22
Заключение 27
Список используемых источников..... 28
Приложение Б. Результаты работы программы. Файл tests.json..... 29
Приложение В. Результаты работы программы. Файл test_results.csv.. 31

Введение

На сегодняшний день информационные технологии играют важную роль в системе образования. Преподаватели высших и средних учебных заведений всё чаще используют программные средства для проверки знаний студентов, особенно в дисциплинах, связанных с программированием. Проверка лабораторных и практических работ, написанных студентами, требует значительных временных затрат, так как преподавателю необходимо запустить каждую программу, проверить её работоспособность на различных наборах входных данных и сравнить полученные результаты с эталонными.

Особую сложность представляет проверка программ, которые обрабатывают входные данные и выводят результат. Преподавателю приходится вручную вводить тестовые данные, анализировать вывод и выставлять оценку. При большом количестве студентов этот процесс становится трудоёмким и подверженным ошибкам, связанным с человеческим фактором. Кроме того, ручная проверка не позволяет проводить регулярное тестирование в процессе обучения.

Решением данной проблемы является автоматизированная система тестирования, которая позволяет преподавателю один раз подготовить набор тестов для каждого задания, после чего проверка работ студентов будет выполняться автоматически. Преподаватель загружает программу студента, система запускает её с подготовленными тестовыми данными и выдаёт отчёт о прохождении тестов.

В связи с этим актуальной задачей является разработка простой и удобной программы для автоматизированного тестирования студенческих работ. Данная курсовая работа посвящена созданию такого приложения — системы «ModulTest», предназначенной для использования в учебных заведениях с целью облегчения работы преподавателей.

Постановка задачи

Необходимо разработать программу для автоматизированного тестирования исполняемых файлов. Программа должна иметь удобный и интуитивно понятный графический интерфейс, позволяющий пользователю быстро выполнять основные действия с тестами: добавление, редактирование, удаление, поиск, сортировку, запуск тестирования, сохранение и загрузку наборов тестов.

В ходе разработки необходимо определить структуру данных, используемых в программе. Для этого следует выделить основные сущности: тест, менеджер тестов, главное окно, диалог добавления и редактирования, окно результатов. Для каждой сущности необходимо определить набор полей, методов и связей. Тест должен содержать уникальный идентификатор, название, описание, входные данные, ожидаемый вывод, дату создания, тип проверки и имя переменной для проверки. Менеджер тестов должен обеспечивать загрузку и сохранение тестов в файл, добавление, удаление и обновление записей, а также поиск и сортировку.

Программа должна быть разделена на логические модули. Многомодульность позволит упростить разработку, отладку и дальнейшее расширение функционала. Отдельные модули должны отвечать за хранение данных, работу с тестами, графический интерфейс, управление файлами и выполнение тестирования.

Необходимо использовать сложные типы данных, такие как классы, списки объектов и файлы. Списки объектов должны применяться для хранения тестов во время работы программы. Файлы должны использоваться для сохранения набора тестов на диск и последующей загрузки ранее сохранённой информации. Формат файлов должен быть JSON для удобства чтения и редактирования.

Режим работы программы должен быть графическим. Для реализации интерфейса необходимо использовать библиотеку Qt Framework. Главная

форма должна отображать список тестов в виде таблицы, а также предоставлять пользователю доступ ко всем основным функциям программы: добавление, редактирование, удаление, запуск тестирования, просмотр результатов, поиск и сортировка.

Устройствами ввода-вывода являются клавиатура, мышь и экран монитора. С помощью клавиатуры пользователь вводит данные о тестах и поисковые запросы. С помощью мыши выполняется выбор пунктов меню, нажатие кнопок, выбор записей в таблицах и открытие форм редактирования.

Интерфейс должен быть построен на основе панели инструментов, полей поиска и таблицы. Это необходимо для создания понятной структуры программы и удобной навигации. Пользователь должен иметь возможность работать с тестами, управлять базой тестов, настраивать тестируемую программу и просматривать результаты.

Также необходимо реализовать поддержку двух типов проверки результатов. При проверке полного вывода программа сравнивает весь вывод тестируемой программы с эталонной строкой. При проверке переменной программа ищет в выводе строку, содержащую имя переменной и знак равенства или двоеточие, и извлекает значение для сравнения.

В результате выполнения задачи должна быть создана система автоматизированного тестирования, предназначенная для проверки корректности работы исполняемых файлов. Программа должна обеспечивать надёжную работу с тестами, удобный пользовательский интерфейс и возможность сохранения информации для дальнейшего использования.

Выбор решения

При запуске программы открывается главное окно приложения. На данном этапе пользователь может создать новый набор тестов, загрузить ранее сохранённый набор из файла или сразу приступить к добавлению тестов. Главное окно содержит таблицу тестов, панель инструментов с кнопками управления, поля для поиска и сортировки.

Программа имеет графический интерфейс и состоит из нескольких основных режимов работы:

– Создать новый набор тестов. Пользователь может начать работу с пустым набором тестов. При необходимости текущие тесты можно сохранить в файл перед очисткой.

– Загрузить набор тестов из файла. Пользователь выбирает файл с ранее сохранёнными тестами. Программа считывает данные из выбранного JSON-файла, восстанавливает записи тестов и отображает их в главном окне.

– Добавить тест. Пользователь нажимает кнопку добавления теста. Открывается отдельная форма, в которой необходимо ввести название, описание, входные данные, ожидаемый результат, выбрать тип проверки и при необходимости указать имя переменной. После подтверждения данные сохраняются в списке тестов, а таблица на главной форме обновляется.

– Редактировать тест. Пользователь дважды нажимает на нужную запись в таблице тестов. Открывается форма редактирования, в которую автоматически подставляются текущие данные выбранного теста. После изменения полей и подтверждения программа обновляет запись в базе тестов.

– Удалить тест. Пользователь выбирает запись в таблице тестов и нажимает кнопку удаления. Программа определяет уникальный идентификатор выбранного теста и удаляет соответствующую запись из списка. После удаления таблица обновляется.

– Запустить тестирование. Пользователь предварительно указывает путь к тестируемой программе через пункт меню «Загрузить проверяемую

программу…». После нажатия кнопки запуска программа последовательно выполняет все тесты. Для каждого теста запускается тестируемая программа, передаются входные данные, считывается вывод и сравнивается с ожидаемым. Программа отображает ход выполнения в строке состояния.

– Просмотреть результаты. После выполнения тестов пользователь может открыть окно результатов. В этом окне отображается таблица с детальной информацией о каждом тесте: идентификатор, название, статус прохождения, ожидаемое значение, фактическое значение и время выполнения. Также отображается сводная статистика: общее количество тестов, количество пройденных, количество непройденных и процент успешности.

– Поиск тестов. Пользователь вводит текст в поле поиска. Программа сравнивает введённую строку с названием и описанием каждого теста. Если совпадения найдены, в таблице отображаются только подходящие записи.

– Сортировка тестов. Пользователь выбирает вариант сортировки из выпадающего списка. Доступна сортировка по названию в алфавитном порядке (A-Z), по названию в обратном порядке (Z-A), по дате создания (сначала новые) и по дате создания (сначала старые).

– Сохранить набор тестов в файл. Пользователь нажимает кнопку сохранения в меню. Программа формирует JSON-представление всех тестов и записывает его в выбранный пользователем файл.

– Настройка тестируемой программы. Пользователь выбирает пункт меню «Загрузить проверяемую программу…». В диалоговом окне необходимо указать путь к исполняемому файлу, который будет тестироваться. Путь сохраняется в настройках и используется при последующих запусках тестов.

– Выход из программы. Пользователь закрывает главное окно программы стандартным способом. При необходимости данные могут быть заранее сохранены в файл.

Для реализации выбранного решения используется многомодульная структура. Классы Test и TestManager отвечают за описание объектов тестов и хранение списка. Классы AddTestDialog и ResultsWindow обеспечивают взаимодействие пользователя с программой при добавлении тестов и просмотре результатов. Класс MainWindow реализует основную логику приложения, включая выполнение тестов через QProcess.

Навигация в программе осуществляется с помощью мыши и клавиатуры. Мышь используется для выбора кнопок, пунктов меню и записей в таблицах, а клавиатура — для ввода данных и поисковых запросов. Такой подход делает программу удобной и понятной для пользователя.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, каждый из которых выполняет определённую функцию. Реализация программы выполнялась поэтапно, от создания основной структуры приложения до разработки отдельных форм, классов данных и функций сохранения информации.

Основные модули программы:

– main.cpp — основной файл программы. Содержит точку входа в приложение. При запуске программы создаётся объект QApplication и открывается главное окно MainWindow.

– test_manager.h и test_manager.cpp — модуль управления тестами. Отвечает за хранение списка тестов, загрузку и сохранение тестов в JSON-файл, а также за добавление, удаление, обновление, поиск и сортировку записей. Содержит структуру Test, в которой определены поля теста, и класс TestManager, реализующий основные операции с базой тестов.

– add_test_dialog.h и add_test_dialog.cpp — модуль добавления и редактирования тестов. Содержит форму для ввода данных о тесте: название, описание, входные данные, ожидаемый результат, тип проверки и имя переменной. Если форма открывается для редактирования, поля автоматически заполняются существующими данными выбранного теста.

– results_window.h и results_window.cpp — модуль отображения результатов. Содержит форму для просмотра результатов тестирования. В форме отображается таблица с детальной информацией о каждом тесте, сводная статистика, а также кнопки для обновления, очистки и экспорта результатов.

– main_window.h и main_window.cpp — главный модуль пользовательского интерфейса. На данной форме отображается таблица тестов. Здесь реализованы основные действия пользователя: добавление, редактирование, удаление, поиск, сортировка, запуск тестирования и

сохранение тестов. Также здесь реализована логика выполнения тестов с использованием QProcess.

– fast_find.h и fast_find.asm — модуль ассемблерных оптимизаций. Содержит функции, реализованные на языке ассемблера для оптимизации критических участков кода. В частности, функция fastFindEqual выполняет быстрый поиск символа «=» в строке с использованием регистров процессора.

Разработка программы выполнялась в следующем порядке:

– Создание основного проекта Qt Widgets Application.

– Реализация точки входа в файле main.cpp.

– Создание главных классов данных Test и TestManager.

– Разработка главной формы MainWindow.

– Реализация формы добавления и редактирования тестов AddTestDialog.

– Создание формы отображения результатов ResultsWindow.

– Реализация функций сохранения и загрузки тестов в JSON-файл.

– Реализация механизма выполнения тестов через QProcess.

– Добавление поиска и сортировки тестов.

– Реализация проверки переменной с извлечением значения из вывода.

– Настройка отображения данных в таблице.

– Написание ассемблерного модуля для оптимизации.

– Проверка работы программы и исправление ошибок.

По порядку реализации модулей можно сделать вывод, что при разработке использовался нисходящий подход. Сначала была спроектирована общая структура программы и основные компоненты, затем постепенно реализовывались отдельные формы, классы и вспомогательные функции.

Такой подход позволил упростить разработку и тестирование, так как каждый модуль можно было проверять сразу после его создания. Благодаря разделению программы на отдельные логические части повысилась

читаемость кода, упростился поиск ошибок и появилась возможность дальнейшего расширения функционала программы.

Описание программы

Приложение ModulTest.exe является основным исполняемым модулем программы. При запуске открывается главное окно, предназначенное для работы с тестами. В главном окне пользователь может добавлять, редактировать, удалять, искать, сортировать тесты, а также выполнять тестирование и просматривать результаты.

В главном окне пользователь может добавлять, редактировать, удалять, искать и фильтровать записи, а также сохранять базу данных в файл. Управление программой осуществляется с помощью мыши и клавиатуры (Таблица 1).

Таблица 1 — Основные элементы главного окна

Элемент, вызывающий событие	Действие пользователя	Действие программы
Кнопка «Добавить тест»	Пользователь выбирает добавление нового теста	Открывается форма AddTestDialog для ввода данных теста
Двойной щелчок по тесту	Пользователь выбирает запись теста	Открывается форма редактирования выбранного теста
Кнопка «Удалить тест»	Пользователь выбирает тест и нажимает кнопку удаления	Программа удаляет запись из списка тестов и обновляет таблицу
Кнопка «Запустить»	Пользователь запускает тестирование	Программа последовательно выполняет все тесты через QProcess
Кнопка «Результаты»	Пользователь открывает результаты	Открывается форма ResultsWindow с детальной статистикой
Поле «Поиск»	Пользователь вводит текст для поиска	Программа отображает только тесты, соответствующие запросу
Список «Сортировка»	Пользователь выбирает вариант сортировки	Программа сортирует тесты по выбранному критерию

При открытии формы в режиме добавления все поля ввода пусты, а при редактировании — автоматически заполняются данными выбранного теста. Пользователь может выбрать тип проверки: полное совпадение вывода или проверка значения переменной (Таблица 2).

Таблица 2 — Работа диалога добавления и редактирования теста

Элемент, вызывающий событие	Действие пользователя	Действие программы
Открытие формы	Пользователь нажимает «Add Test»	Программа открывает пустую форму для ввода данных
Открытие формы с выбранным ID	Пользователь дважды нажимает на запись теста	Программа заполняет поля формы данными выбранного теста
Поля ввода	Пользователь вводит название, описание, входные данные и ожидаемый результат	Программа временно хранит введённые значения в элементах формы
Radio «Check only variable»	Пользователь выбирает проверку переменной	Программа активирует поле ввода имени переменной
Кнопка подтверждения	Пользователь подтверждает ввод	Программа создаёт новый тест или изменяет существующую запись

Процесс выполнения тестов полностью автоматизирован (Таблица 3). Программа последовательно запускает тестируемый файл, передаёт ему входные данные, считывает вывод и сравнивает с эталоном. При зависании тестируемой программы срабатывает таймаут.

Таблица 3 — Выполнение тестирования

Этап	Действие программы
1	Программа запускает тестируемый исполняемый файл через QProcess::start
2	Программа ожидает запуска процесса с таймаутом 3 секунды
3	Программа передаёт входные данные через QProcess::write
4	Программа закрывает канал записи через QProcess::closeWriteChannel
5	Программа ожидает завершения процесса с таймаутом 5 секунд
6	Программа считывает вывод через QProcess::readAllStandardOutput
7	При необходимости программа извлекает значение переменной из вывода
8	Программа сравнивает полученный результат с ожидаемым
9	Программа сохраняет результат в список и переходит к следующему тесту

Пользователь может просматривать детальную информацию о каждом тесте, обновлять данные, очищать историю и экспортировать результаты в файлы формата CSV или JSON (Таблица 4).

Таблица 4 — Окно результатов

Элемент, вызывающий событие	Действие пользователя	Действие программы
Открытие окна	Пользователь нажимает «Results»	Программа загружает результаты из файла и отображает их в таблице
Кнопка «Refresh»	Пользователь обновляет результаты	Программа перезагружает данные из файла и обновляет таблицу
Кнопка «Clear All»	Пользователь очищает результаты	Программа удаляет файл с результатами и очищает таблицу
Кнопка «Export Results»	Пользователь экспортирует результаты	Программа сохраняет результаты в CSV или JSON-файл

Программа поддерживает сохранение набора тестов в JSON-файл и загрузку ранее сохранённых тестов (Таблица 5). Также предусмотрена настройка пути к тестируемой программе, которая сохраняется в параметрах приложения.

Таблица 5 — Работа с файлами тестов

Элемент, вызывающий событие	Действие пользователя	Действие программы
Меню «Save tests»	Пользователь сохраняет тесты	Программа вызывает метод TestManager::saveToFile и создаёт JSON-файл
Меню «Load tests»	Пользователь загружает тесты	Программа отображает диалог выбора файла и вызывает TestManager::loadFromFile
Меню «Configure executable»	Пользователь настраивает тестируемую программу	Программа отображает диалог выбора исполняемого файла и сохраняет путь в настройках

Таким образом, программа представляет собой графическое приложение. Все основные действия выполняются через формы, кнопки и таблицы, что делает работу с программой удобной и понятной для пользователя.

На рисунке 1 представлена схема взаимодействия программы.

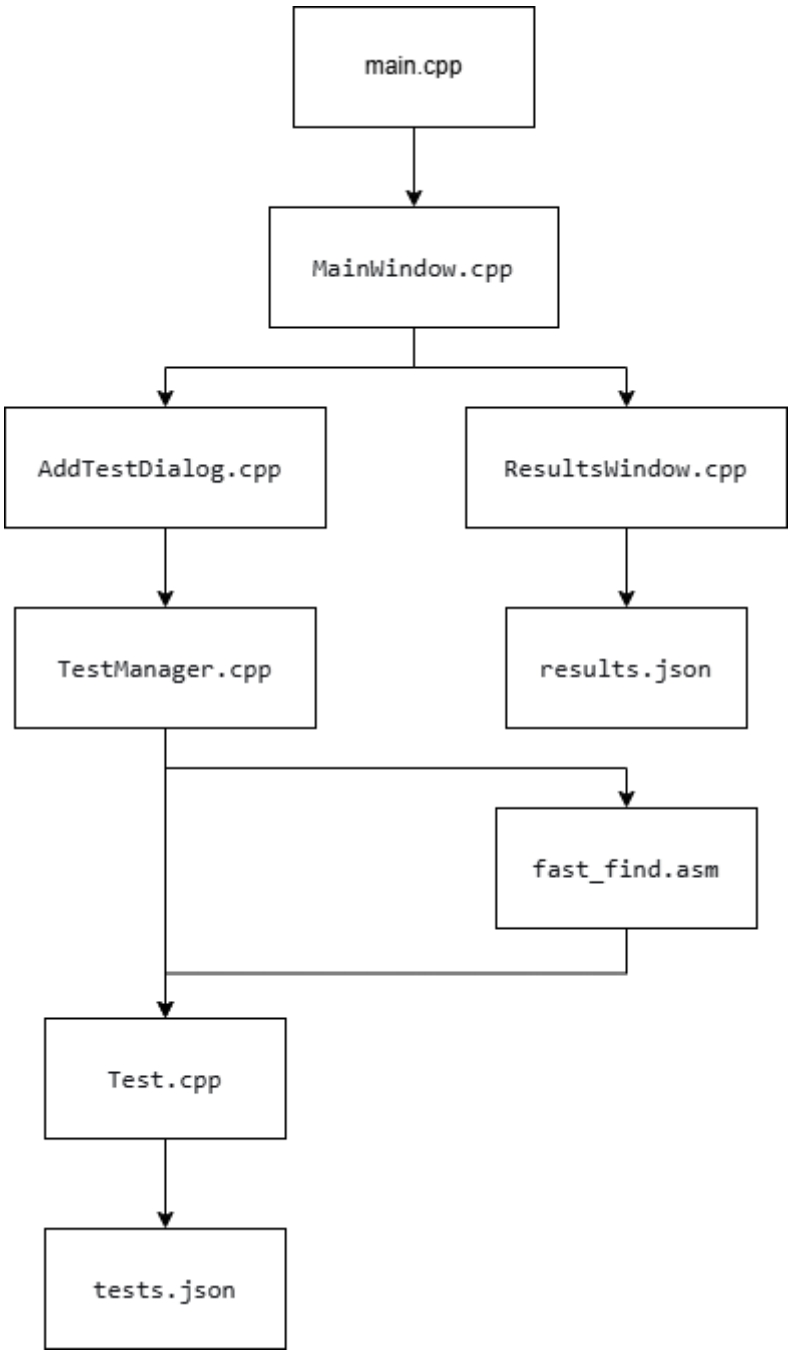

Рисунок 1 - Схема взаимодействия программы

На рисунке 2 представлена схема данных.

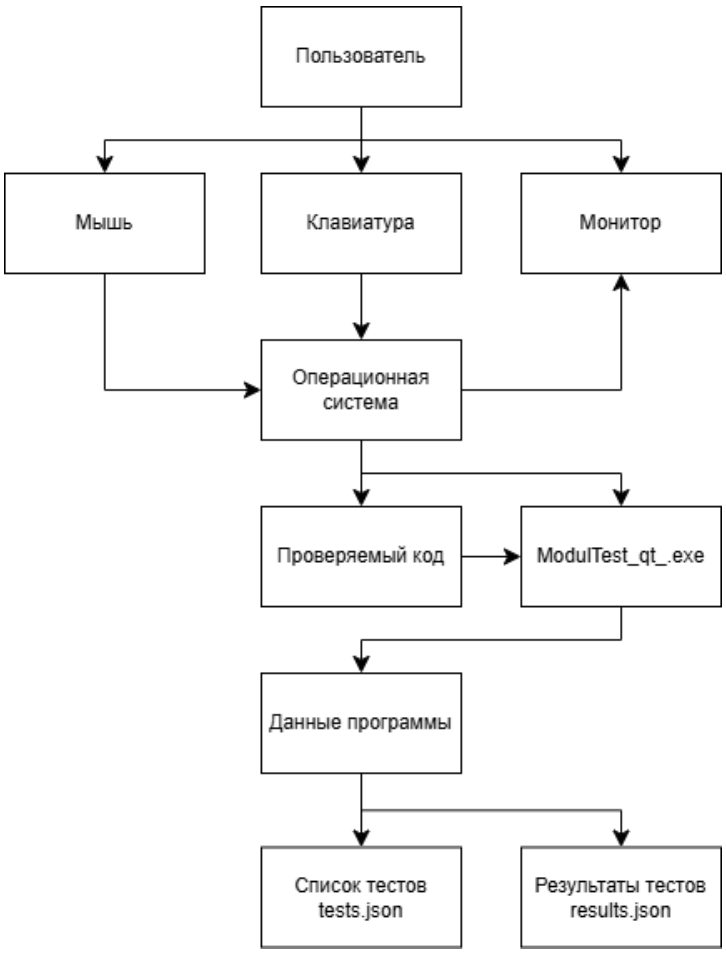

Рисунок 2 - Схема данных

Руководство пользователя

Программа ModulTest.exe предназначена для автоматизированного тестирования исполняемых файлов. Программа имеет удобный графический интерфейс и поддерживает такие операции, как создание нового набора тестов, загрузка тестов из файла, добавление записей, удаление записей, изменение записей, поиск тестов, сортировка тестов по названию или дате, запуск тестирования и просмотр результатов.

После запуска программы открывается главное окно (Рисунок 3). В данном окне пользователь может добавлять, редактировать и удалять тесты, а также выполнять их.

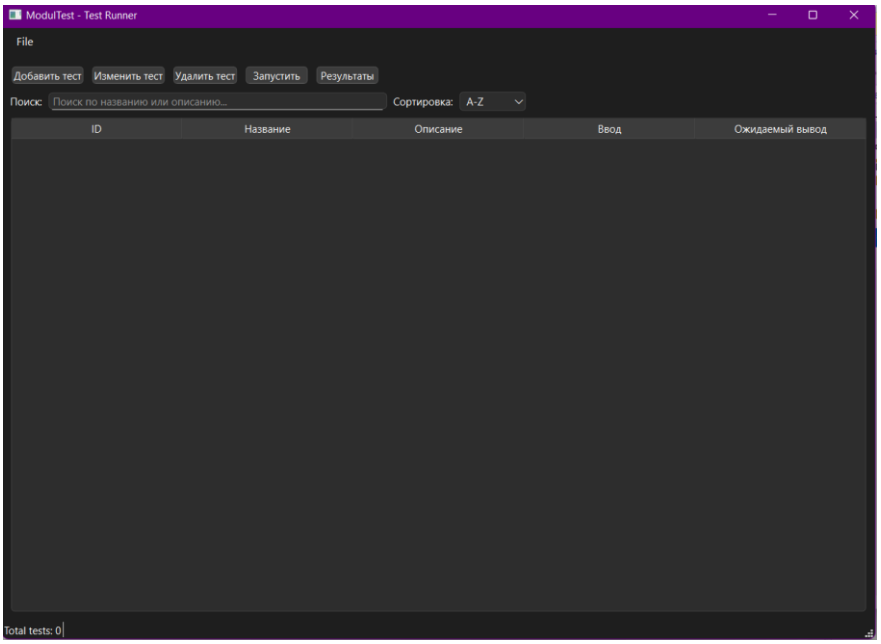

Рисунок 3 - Главное окно

Для настройки тестируемой программы необходимо выбрать пункт меню «File» → «Загрузить проверяемую программу» (Рисунок 4). В диалоговом окне нужно указать путь к исполняемому файлу, который будет тестироваться.

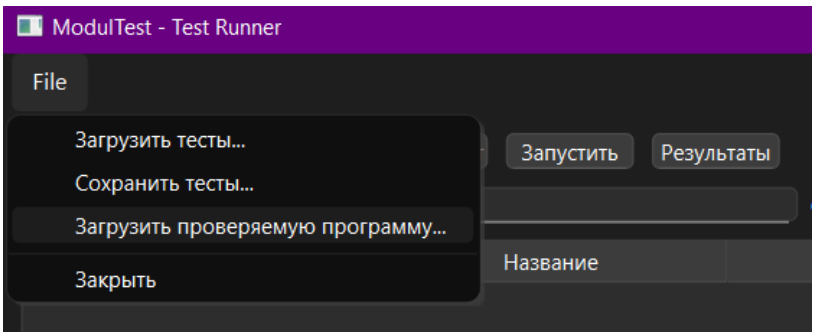

Рисунок 4 - Меню File

Для добавления нового теста необходимо нажать кнопку «Добавить тест». После этого откроется форма ввода данных (Рисунок 5). В ней нужно заполнить название, описание, входные данные и ожидаемый результат. Также необходимо выбрать тип проверки. Если выбран тип «Проверить только переменную», нужно указать имя переменной. После заполнения всех полей необходимо нажать кнопку подтверждения. Новая запись появится в таблице тестов.

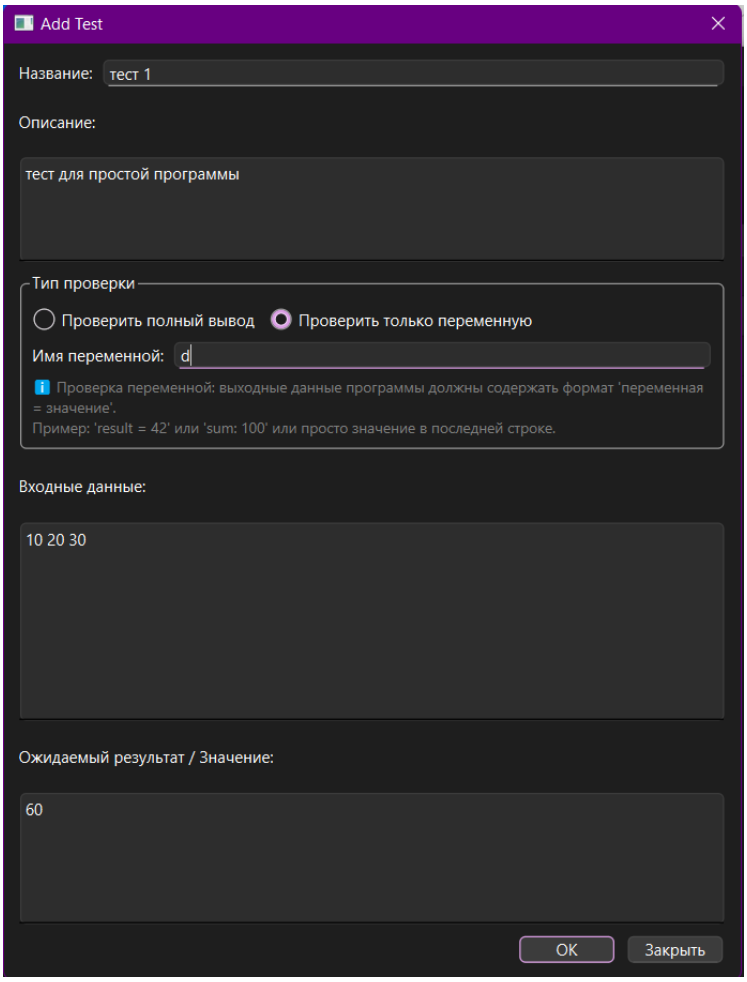

Рисунок 5 - Форма ввода данных

Для изменения данных теста необходимо дважды щёлкнуть по нужной записи в таблице тестов. Откроется форма редактирования, в которой будут указаны текущие данные выбранного теста. Пользователь может изменить нужные поля и подтвердить изменения. После этого запись в таблице будет обновлена.

Для удаления теста необходимо выбрать нужную запись в таблице тестов и нажать кнопку «Удалить тест». Программа удалит выбранную запись из базы тестов и обновит таблицу.

Для запуска тестирования необходимо нажать кнопку «Запустить». Программа последовательно выполнит все тесты, отображая ход выполнения в строке состояния (Рисунок 6).

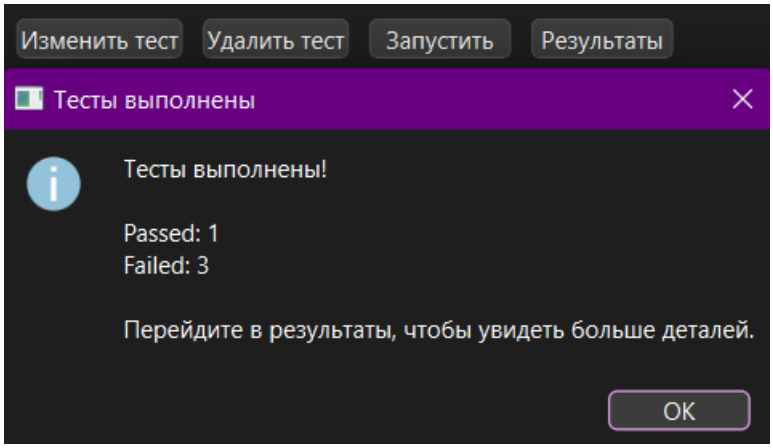

Рисунок 6 - Ход выполнения

Для просмотра результатов необходимо нажать кнопку «Результаты». После этого откроется окно результатов (Рисунок 7), в котором отображаются детальные сведения о прохождении каждого теста, а также сводная статистика. Пройденные тесты отмечены зелёным цветом, непройденные — красным.

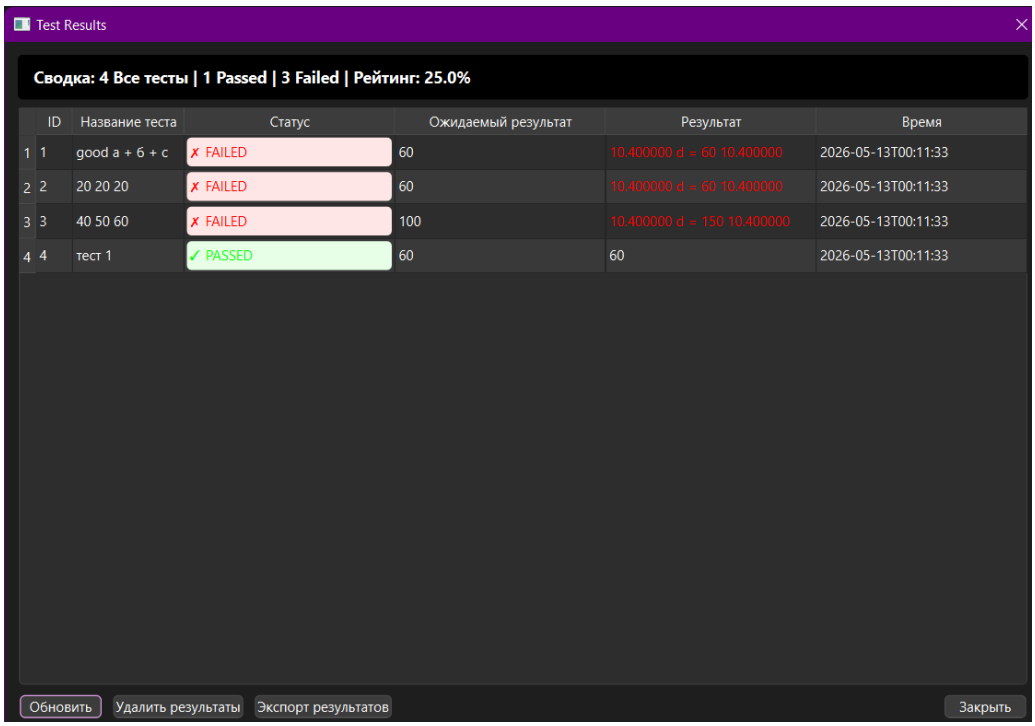

Рисунок 7 - Окно результатов

Для поиска теста необходимо ввести название или описание в поле поиска. Программа автоматически отобразит только те записи, которые соответствуют введённому запросу. Если очистить поле поиска, в таблице снова будут отображены все тесты.

Для сортировки тестов необходимо выбрать нужный вариант из выпадающего списка «Сортировка:». Доступны следующие варианты: A-Z (сортировка по названию по возрастанию), Z-A (сортировка по названию по убыванию), «Новее» (сортировка по дате создания, сначала новые), «Старше» (сортировка по дате создания, сначала старые).

Для сохранения набора тестов необходимо выбрать меню «File» → «Сохранить тесты». Программа откроет диалог выбора места сохранения и создаст JSON-файл с текущими тестами.

Для загрузки набора тестов необходимо выбрать меню «File» → «Загрузить тесты». Программа откроет диалог выбора файла, после чего загрузит тесты из выбранного JSON-файла и отобразит их в таблице.

Для выхода из программы необходимо закрыть главное окно стандартной кнопкой закрытия. Перед выходом рекомендуется сохранить набор тестов, чтобы внесённые изменения не были потеряны.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульного программного приложения на языке C++ с использованием фреймворка Qt. Была изучена работа с классами, объектами, списками, файлами в формате JSON и графическим интерфейсом Qt Widgets. Также были получены практические навыки проектирования структуры программы, обработки пользовательского ввода, сохранения и загрузки данных, а также отладки и тестирования программы в среде Visual Studio 2022 на языке C++ с элементами ассемблера.

В рамках курсовой работы была разработана программа «ModulTest» для автоматизированного тестирования исполняемых файлов. Программа имеет все необходимые функции для ведения базы тестов: создание нового набора, загрузку данных из файла, добавление, редактирование и удаление записей, поиск и сортировку тестов, а также выполнение тестирования с передачей входных данных и анализом вывода.

Разработанная программа имеет удобный графический интерфейс, что упрощает взаимодействие пользователя с системой. Использование таблиц, кнопок, полей ввода и выпадающих списков делает работу с тестами понятной и наглядной.

В дальнейшем программу можно улучшить путём добавления возможности группового выполнения тестов, поддержки многопоточного выполнения для ускорения тестирования, расширенной проверки переменных с использованием регулярных выражений, а также интеграции с системами непрерывной интеграции.

Список используемых источников

1. Microsoft Learn. Документация по языку C++ [Электронный ресурс]. -- 2025. -- URL: https://learn.microsoft.com/ru-ru/cpp/ (дата обращения: 01.05.2026).
2. Qt Documentation. Qt 6.8 Reference Documentation [Электронный ресурс]. -- The Qt Company, 2025. -- URL: https://doc.qt.io/ (дата обращения: 01.05.2026).
3. Microsoft Learn. Документация по MASM для x64 (ml64.exe) [Электронный ресурс]. -- Microsoft Corporation, 2024. -- URL: https://learn.microsoft.com/ru-ru/cpp/assembler/masm/ (дата обращения: 01.05.2026).
4. Schildt H. C++: The Complete Reference. -- 4th ed. -- New York: McGraw-Hill, 2022. -- 1056 p.
5. Vandevoorde D., Josuttis N., Gregor D. C++ Templates: The Complete Guide. -- 2nd ed. -- Boston: Addison-Wesley, 2024. -- 832 p.
6. Stroustrup B. A Tour of C++. -- 3rd ed. -- Boston: Addison-Wesley, 2022. -- 256 p.
7. Meyers S. Effective Modern C++. -- Sebastopol: O'Reilly Media, 2024. -- 320 p.
8. Williams A. C++ Concurrency in Action. -- 2nd ed. -- Shelter Island: Manning Publications, 2023. -- 592 p.
9. Josuttis N. C++17 - The Complete Guide. -- Independently published, 2022. -- 450 p.

Приложение Б. Результаты работы программы. Файл tests.json


```
[
  {
    "created": "2026-04-29T12:08:02",
    "description": "",
    "expectedOutput": "60",
    "id": 1,
    "inputData": "10 20 30",
    "name": "good a + б + c",
    "validationType": 0,
    "variableName": ""
  },
  {
    "created": "2026-04-29T12:08:17",
    "description": "",
    "expectedOutput": "60",
    "id": 2,
    "inputData": "20 20 20",
    "name": "20 20 20",
    "validationType": 1,
    "variableName": "d"
  },
  {
    "created": "2026-04-29T12:08:35",
    "description": "",
    "expectedOutput": "100",
    "id": 3,
    "inputData": "40 50 60",
    "name": "40 50 60",
```

```
        "validationType": 0,
        "variableName": ""
    }
]
```

Приложение В. Результаты работы программы. Файл test_results.csv

ID,Test Name,Status,Expected,Actual,Timestamp

1,"good a + б + c",FAILED,"60","10.400000 d = 60 10.400000",2026-05-27T12:48:21

2,"20 20 20",PASSED,"60","60",2026-05-27T12:48:21

3,"40 50 60",FAILED,"100","10.400000 d = 150 10.400000",2026-05-27T12:48:21