УДК 621.316

RIGIDBUSBARS – КОМПЬЮТЕРНАЯ ПРОГРАММА РАСЧЕТА ЭЛЕКТРОДИНАМИЧЕСКОЙ СТОЙКОСТИ ЖЕСТКОЙ ОШИНОВКИ С ПРОИЗВОЛЬНОЙ ОРИЕНТАЦИЕЙ ШИН И ИЗОЛЯТОРОВ

Шпаковский А.А., Баран А.Г.
Научный руководитель – Климкович П.И.

Программа RigidBusbars является логическим продолжением программы EDU. Причиной создания нового проекта стала неверная архитектура приложения, заложенная на этапе проектирования, что сделало нецелесообразным внесение дополнительного функционала. Сложилась такая ситуация, что для внесения очередной правки каждый раз требовалось переписывать большой участок кода.

Поэтому было решено заново написать программу в соответствии с объектно-ориентированной парадигмой программирования. Однако одно лишь использование ООП само по себе не является решением проблемы. И даже может при неверном использовании усложнить программу. Эта проблема является широко известной, поэтому на этапе создания архитектуры приложения принято использовать паттерны проектирования.

Основываясь на опыте, полученном при создании первой версии программы, было решено:

– Использовать концепции объектно-ориентированного программирования (ООП). Что позволило составить программу из мало зависящих друг от друга частей, т. о. изменения в одной ее части не влияют на всю программу целиком. Наиболее значимым и заметным результатом стало разделение визуального интерфейса и расчетной части кода программы. Также следует заметить, что из-за слабой связи компонентов программы между собой появилась возможность повторного использования кода, в других проектах.

– Использовать классические паттерны проектирования, что облегчает организацию взаимодействия между различными частями программы. Так использование MVC (model view controller) дало возможность простой организации интерактивного интерфейса.

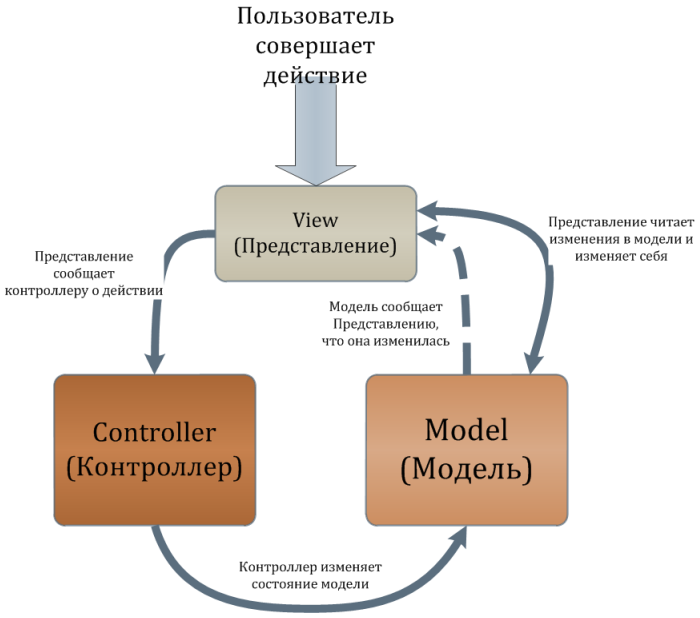

Рисунок 1 – Модель-представление-контроллер

Сложности, возникающие при начальной организации архитектуры в соответствии с шаблонами проектирования, компенсируются в дальнейшем удобством и простотой внедрения дополнительного функционала.

В качестве среды разработки была выбрана интегрированная среда разработки программного обеспечения – Visual Studio 2010 (позднее был осуществлен переход на Visual Studio 2012). Следует заметить, что использовалась лицензионная копия Visual Studio, полученная в рамках проекта Dreamspark от Microsoft.

При работе в Visual Studio доступны три основных способа создания интерактивных приложений Windows.

– Использование интерфейса API Windows. Это базовый интерфейс, предоставляемый операционной системой для взаимодействия с приложением, выполняющимся под ее управлением.

– Использование Windows Forms. Это основанный на формах механизм разработки для создания приложений, выполняющихся под управлением среды CLR.

– Использование классов Microsoft Foundation Classes, известных также как библиотека MFC. Это набор классов на языке C++, инкапсулирующих интерфейс API Windows.

Для разработки программы была выбрана библиотека MFC, т. к. при ее использовании упрощается обращение к функциям ОС. Windows Forms в целом не получила признания среди программистов, поэтому данный способ изначально был отвергнут.

К недостаткам использования библиотеки MFC необходимо отнести большой размер конечной программы, однако в нашем случае это не является решающим.

Главной задачей при разработке графической оболочки программы является создание интуитивно понятного интерфейса, т. е. понятного для человека впервые запустившего программу. В первоначальных версиях проекта (рисунок 2) для этого был принят ряд мер, таких как:

– подробный раздел «Помощь»;

– вывод поясняющего текста в части активного окна;

– стандартное для windows-программ расположение элементов управления.

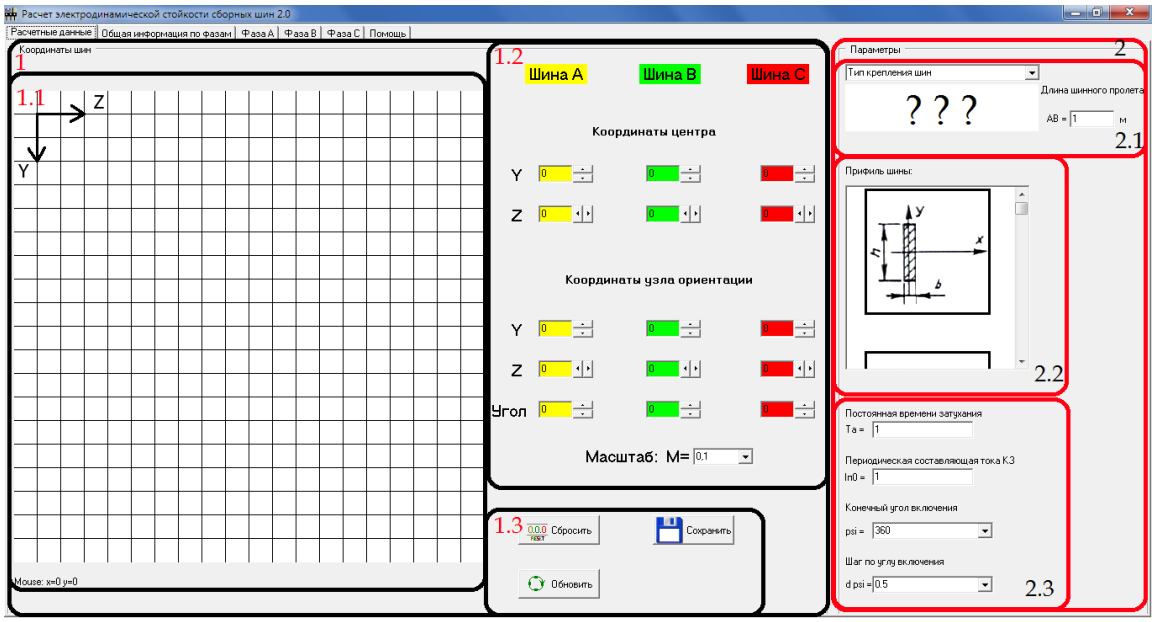

Рисунок 2 – Изначальный дизайн программы

Однако практика показала, что эти меры решают данную проблему не в полной мере. Раздел «Помощь» обычно игнорируется пользователями, что делает его бесполезным. Вывод контекстно-зависимого поясняющего текста является более эффективным, т. к. его отображение производится во время всего процесса работы с программой. Однако этот метод имеет свои недостатки, например, такие как загромождение окон программы и визуальное их усложнение. Использование же стандартного расположения элементов управления это скорее рекомендация для недопущения необоснованного роста сложности графической оболочки.

Принимая во внимание недостатки примененных ранее методов, было решено реализовать взаимодействие пользователя с программой в виде Мастера (Мастер – интерактивная функция в графическом интерфейсе пользователя, которая представляет собой последовательно сменяющие друг друга диалоговые окна для выполнения определённой задачи, которую можно разбить на этапы). Назначение мастера в том, чтобы «провести» пользователя от стартового окна программы до результатов расчета. Приведем логическую структуру интерфейса на рисунке 3.

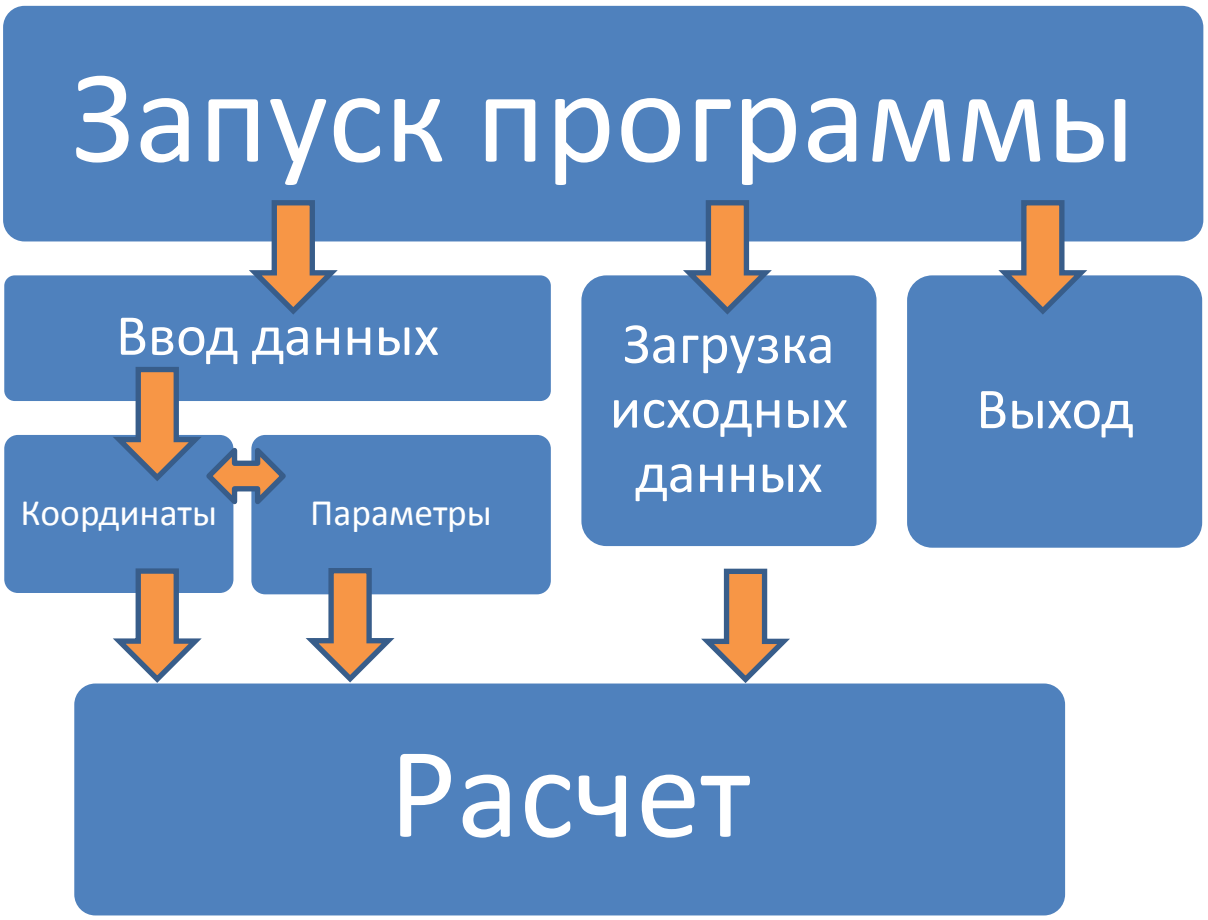

Рисунок 3 – Логическая структура визуального интерфейса

Примером функционального минимализма, которому должна соответствовать вся программа является стартовое окно (рисунок 4). При запуске пользователю предлагается минимум вариантов, кнопки «Ввести данные» и «Загрузить данные» запускают Мастер ввода данных.

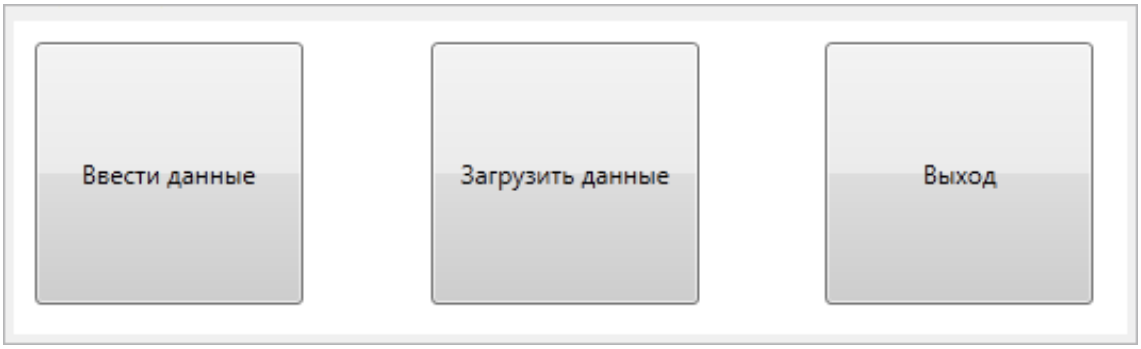

Рисунок 4 – Стартовое окно программы

Т.о. создание «понятного» интерфейса представляет собой некое ограничение пользователя в выборе, отображение минимально необходимой информации и настроек в данный момент времени.

У любого разработчика на стадии проектирования возникает желание как можно больше расширить функционал программы, добавить новые возможности. Зачастую это мотивировано не объективной необходимостью, а простотой реализации. Однако стоит понимать, что хорошо написанная программа это не та программа, которая «все умеет», а которая максимально точно выполняет возложенные на нее задачи.

Однако добавление в программу т. н. «Режима исследования» было необходимо для полноценного исследования электродинамических усилий. Режим исследования становится доступен после выполнения основного расчета.

Данный режим предоставляет возможность более детального расчета для фиксированного угла или момента времени. Также есть возможность проанализировать влияние угла поворота изолятора на возникающие электродинамические усилия.

Изначально планировалось не переписывать имеющийся алгоритм, написанный на языке программирования ForTran, а лишь создать dll (динамически подключаемая библиотека) на его основе и подключить к проекту, но это делало невозможным реализацию режима исследований. Поэтому было решено полностью перенести исходный код на C++. Перенос кода также потребовал проведения тестирования, т. к. даже при простом переводе программы на другой язык программирования могут возникнуть ошибки. Причиной этому могут быть несоответствия типов данных, различные точности основных констант, а также внесенные человеком ошибки. При проведении тестов был замечен незначительный рост скорости выполнения программы.

Рассмотрим последовательность действий при вводе исходных данных при помощи мастера.

Заметным отличием от предыдущей версии является то, что шины уже отображаются на координатном поле, т. е. достаточно выделить мышкой изображение шины и перенести ее на нужное место. При выделении иконки шины рядом с ней появляется поле Edit для ввода угла поворота (рисунок 5).

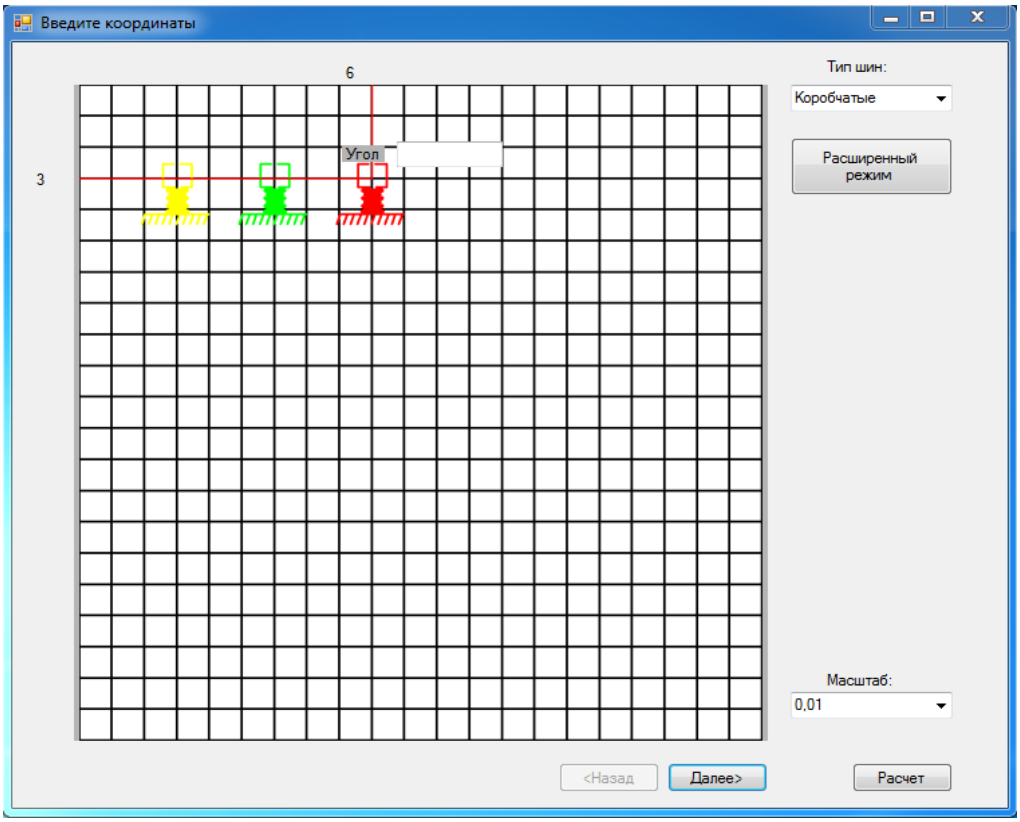

Рисунок 5 – Окно ввода координат мастера ввода данных

Точность ввода графического ввода координат можно повысить или же понизить, изменив масштаб. На рисунке 6 окно ввода координат с масштабом 1 деление = 0,1 метр.

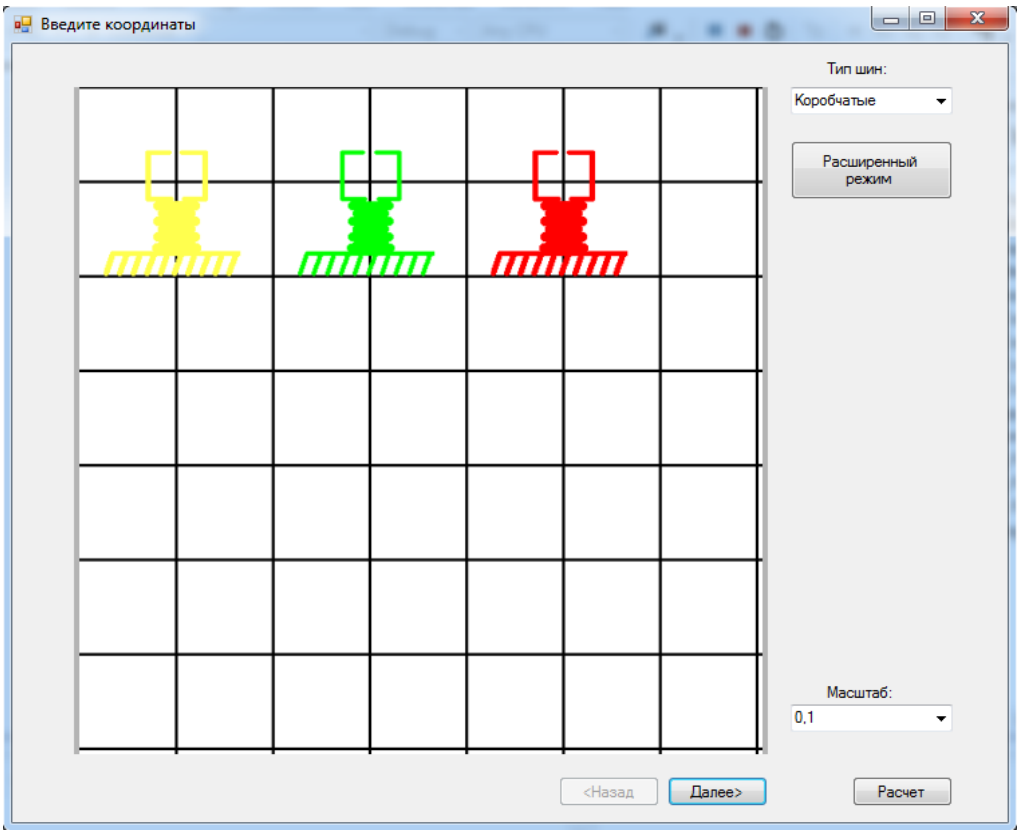

Рисунок 6 – Изменение масштаба

Изменение типа шины динамически изменяет схематический рисунок шин на соответствующий (рисунок 7).

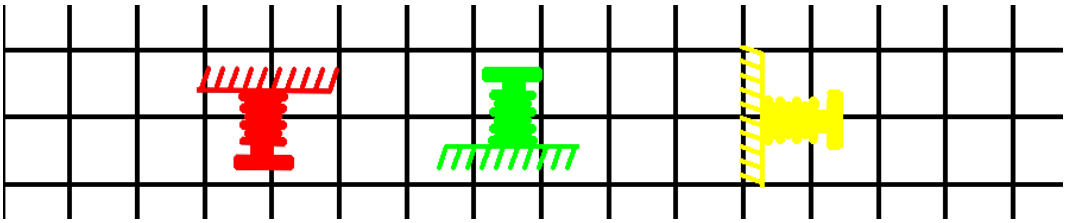

Рисунок 7 – Измененное схематичное изображение шин

Кнопка «Расширенный режим» открывает окно для ручного задания координат. Со временем планируется усовершенствовать графический ввод координат и отказаться от ручного ввода.

Первоначальный вариант окна ввода дополнительных параметров оказался весьма удачным, поэтому изменения коснулись только взаимного расположения элементов (рисунок 8).

Вывод результатов расчета по-прежнему осуществляется в виде текста и графиков (рисунок 9–10). В первой версии программы использовались встроенные классы для создания графиков. Однако функционал стандартных графиков был ограничен, а закрытый исходный код не позволял вносить в них изменения под нужды проекта. Самостоятельное создание графиков весьма трудоемкая задача, сравнимая по сложности с самим проектом. Поэтому единственным решением является найти свободно распространяемую библиотеку с открытым исходным кодом.

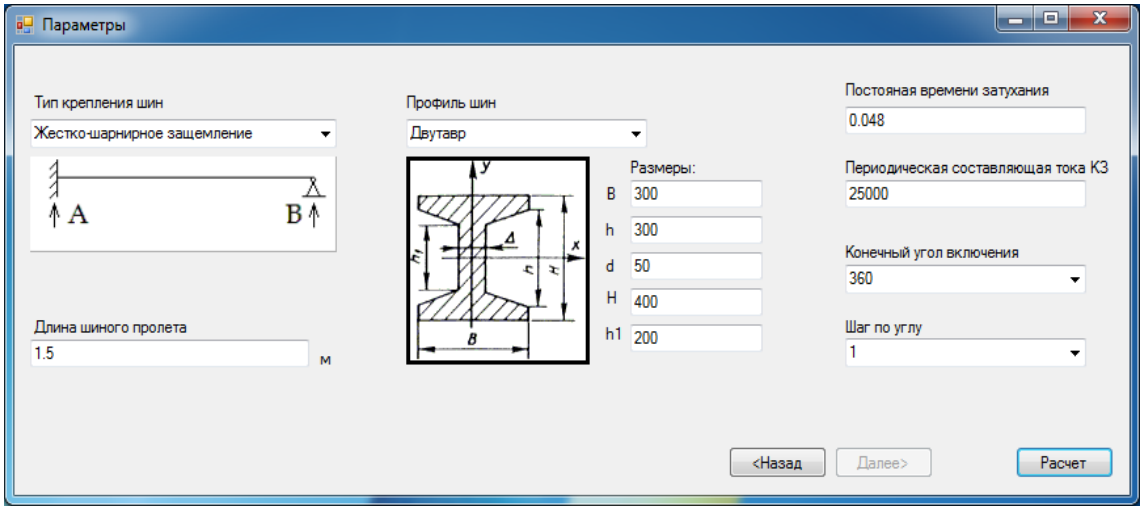

Рисунок 8 – Окно ввода параметров расчета

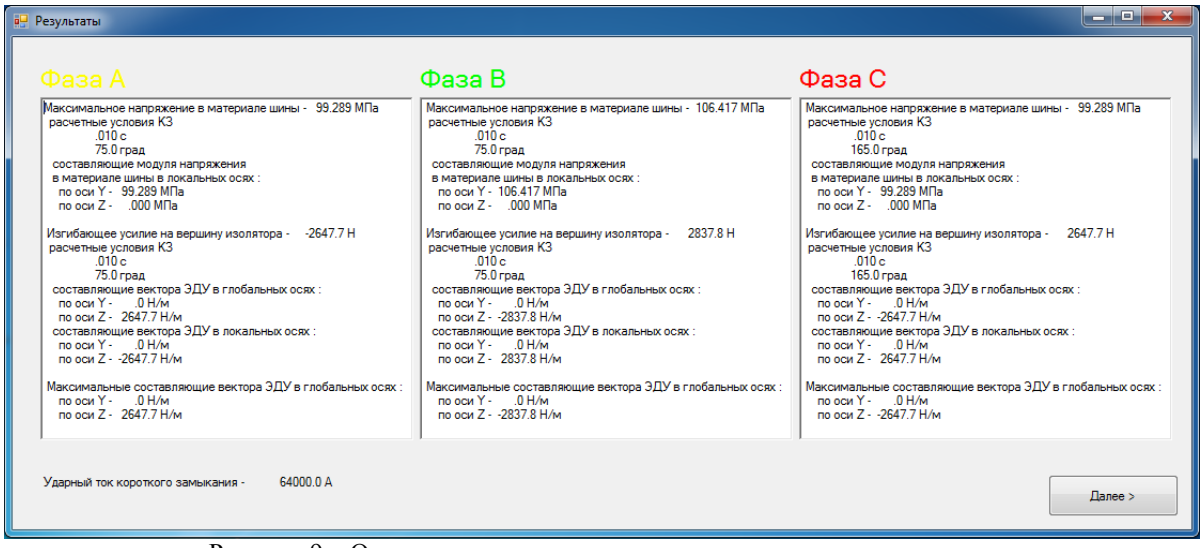

Рисунок 9 – Окно вывода результатов расчета в текстовом виде

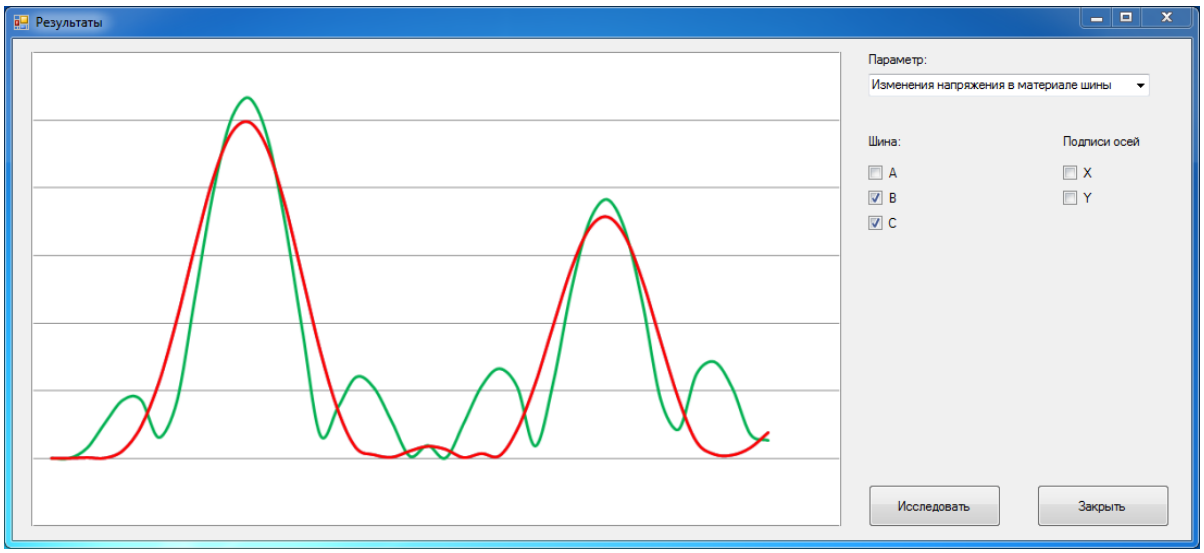

Рисунок 10 – Изменение напряжений в материале шины