Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

 УТВЕРЖДАЮ:

 Зав. кафедрой

 \_\_\_\_\_\_\_\_\_\_ Дворянкин А.М.

 «\_\_\_»\_\_\_\_\_\_\_ 2016 г.

**НАДЕЖНОСТЬ И КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

«Расчет цепи переменного тока»

Описание программы

СОГЛАСОВАНО: Разработчик:

Руководитель работы: Студент ПрИн-266

Доцент кафедры ПОАС \_\_\_\_\_\_\_\_\_\_ Островский Д.С.

\_\_\_\_\_\_\_\_\_\_ Сычев О.А. «\_\_\_»\_\_\_\_\_\_\_ 2016 г.

«\_\_\_»\_\_\_\_\_\_\_ 2016 г.

 Нормоконтролер:

 Ассистент кафедры ПОАС

 \_\_\_\_\_\_\_\_\_\_ Мамонтов Д.П.

 «\_\_\_»\_\_\_\_\_\_\_ 2016 г.

2016

Содержание

1 Общие сведения………………………………………………….……………..3

2 Функциональное назначение……………………………………………….....3

3 Описание логической структуры…………………………………………......3

4 Используемые технические средства…………………………………..…….4

5 Вызов и загрузка………….………………………………….…………………4

6 Входные и выходные данные..………………………………………………..4

Приложение А. Описание структур данных ….……………………………….5

Приложение Б. Описание алгоритмов функций …………..…………………..7

Приложение В. Диаграмма вызовов функций…………………………………21

Приложение Г. Диаграмма потоков данных………………………………..…22

1 Общие сведения

Программа «CalculationRLC» должна рассчитать значения токов в каждой из ветвей цепи на основе полученной цепи и значений ее составляющих. Для корректной работы программы необходима операционная система Windows XP или выше. Дополнительного ПО не требуется.

Программа написана на языке C++ с использованием библиотеки Qt.

2 Функциональное назначение

Данная программа позволяет рассчитать значения токов каждой из ветвей полученной цепи с учетом ее характеристик.

3 Описание логической структуры

Работа программы может быть логически разделена на несколько частей:

* считывание данных из файлов;
* нахождение перечислений;
* записывание данных в файл.

4 Используемые технические средства

Описание требуемых технических средств содержится в документе «Техническое задание», пункт 3.4.

5 Вызов и загрузка

Программа запускается с помощью командной строки с указанием имени файла с расширением .txt, в котором содержится описание электрической цепи в формате xml.

6 Входные и выходные данные

Формат входных и выходных данных описан в документе «Техническое задание», пункт 3.5.

Приложение А

 Описание структур данных

class Circuit

{

 public:

 double e;

 double f;

 struct Branch

 {

 int StartNode;

 int EndNode;

 double Resistance;

 double Inductance;

 double Capacitance;

 }

 QVector<Branch> DescriptionBranch;

};

struct StructError

{

 QString Message;

 int ErrorString;

 int ErrorColumn;

};

QVector<StructError> Error;

QVector<QVector<double>> PotentialSystem;

QVector<double> Currents;

QVector<double> Potentials;

e – значение ЭДС цепи

f – значение частоты тока цепи

Branch – структура, описывающая ветви электрической цепи. Содержит поля:

StartNode – индекс выходного узла ветви

EndNode – индекс входного узла ветви

Resistance – значение сопротивления всех резисторов ветви

Inductance – значение индуктивности всех катушек ветви

Capacitance – значение емкости всех конденсаторов ветви

DescriptionBranch – массив описания всех ветвей цепи

StructError – структура, описывающая ошибки

 Message – сообщение об ошибке

 ErrorString – индекс строки с ошибкой

 ErrorColumn – индекс колонки с ошибкой

Error – массив ошибок

PotentialSystem – система уравнений потенциалов

Currents – массив значений токов всех ветвей

Potentials – массив значений потенциалов узлов

Приложение Б

Описание алгоритмов функций

Главная функция программы

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

Входные данные:

argc – количество переданных аргументов командной строки

argv[] – аргументы командной строки;

 argv[0] – программа для запуска

argv[1] – файл с описанием электрической цепи

Выходные данные:

1 - при успешном выполнении, 0 - в противном случае

Алгоритм функции:

1. Проверить наличие аргумента, который содержит в себе файл с расширением .txt. Если аргумента нет или расширение файла является другим, то записать ошибку в массив.
2. Считать данные из текстового файла и записать их в массив структур, описывающий электрическую цепь (ReadXML)
	1. Если данные из текстового файла успешно считаны, то рассчитать токи во всех ветвях электрической цепи (CalculationRLC)
	2. Вывести значения токов для каждой ветви электрической цепи в текстовый файл (OutputCurrents)
3. Если массив с ошибками не пустой, выбрать по коду в массиве ошибку, и вывести сообщение соответствующее этой ошибке

Функция формирования дерева по содержимому файла в формате xml

bool ReadXML(QString Filename, QVector<StructError> &Error, Circuit& In);

Входные данные:

Filename – имя файла, содержащего описание электрической цепи

Выходные данные:

In – описание электрической цепи

 Error – массив ошибок

true, если считывание успешно, false – если считать не удалось

Алгоритм функции:

1. Открыть файл с описанием электрической цепи
2. Если файл можно прочитать
	1. Если можно установить кодировку документа(setContent)
		1. Получить корневой тег xml файла (documentElement)
		2. Записать содержимое ветвей цепи в контейнер структур ветвей (DocumentToStruct)
		3. Проверить записанные данные на ошибки (Errors)
		4. Если ошибки есть
			1. Вернуть false
	2. Иначе если функция установления кодировки выдала ошибку «unexpected end of file»
		1. Записать ошибку о пустом файле в массив
		2. Закрыть файл
		3. Вернуть false
	3. Иначе
		1. Записать ошибку о нераспознанном символе в массив
		2. Закрыть файл
		3. Вернуть false
	4. Закрыть файл
3. Вернуть true

Функция проверки заполненных данных

bool Errors(int CountBranches, QVector <StructError> &Error, Circuit& In);

Входные данные:

Processing – флаг обработки документа

 CountBranches – количество ветвей цепи

 In – описание электрической цепи

Выходные данные:

Error – массив ошибок

 true – если нет ошибок, false – в обратном случае

Алгоритм функции:

1. Если массив ошибок не пустой
	1. Вернуть false
2. Если значение ЭДС или частоты не было записано
	1. Записать ошибку в массив
	2. Вернуть false
3. Для каждой ветви
	1. Если хотя бы одно значение элементов цепи (кроме источника тока) не равно нулю
		1. Считать элементы введенными
	2. Если хотя бы одно из значений не входит в заданный ему диапазон
		1. Записать ошибку в массив
		2. Вернуть false
4. Если ни одного элемента не введено
	1. Записать ошибку о коротком замыкании в массив
	2. Вернуть false
5. Если ветвей больше одной
	1. Для каждого узла
		1. Для каждой ветви
			1. Если узел является входящим или выходящим
				1. Инкрементировать счетчик связанных с узлом ветвей
		2. Если узел упоминается меньше, чем в трех ветвях
			1. Записать ошибку о связывании узлом некорректного числа ветвей в массив
			2. Вернуть false
6. Вернуть true

Функция формирования дерева по содержимому файла в формате xml

QVector <Circuit::Branch> DocumentToStruct(QDomElement DomElement, int \*CountNodes, QVector <StructError> &Error, Circuit &In);

Входные данные:

DomElement – корневой тег xml документа

Выходные данные:

In – описание электрической цепи

CountNodes – переменная, в которую записывается количество узлов цепи

Error – массив ошибок

 Контейнер структур с описанием ветвей электрической цепи

Алгоритм функции:

1. Найти первого потомка корневого узла (firstChild)
2. Пока есть потомок
	1. Если найденный потомок является элементом (isElement)
		1. Преобразовать узел к элементу (toElement)
		2. Если преобразовать удалось
			1. Обработать его атрибут (ProcessingAttribute)
			2. Если массив с ошибками пустой
				1. Вызвать эту же функцию для выполнения обработки внутри этого узла
				2. Перейти к соседнему потомку

Функция обработки атрибута элемента дерева

void ProcessingAttribute(QDomElement DomElement, QVector <StructError> &Error, Circuit& In);

Входные данные:

 DomElement – рассматриваемый элемент дерева

 In – описание электрической цепи

Выходные данные:

Error – массив ошибок

Алгоритм функции:

1. Если тег элемента «branch»
	1. Запомнить его атрибут «index» как текущий номер ветви
	2. Если атрибут не является числом
		1. Записать ошибку в массив
2. Если тег элемента «resistor»
	1. Записать в текущую ветвь сумму уже имеющегося сопротивления резисторов и сопротивления, описанного в данном элементе (SetNextAttributeValueAsDouble)
	2. Если не удалось записать значение
		1. Записать ошибку в массив
3. Если тег элемента «inductor»
	1. Записать в текущую ветвь сумму уже имеющегося сопротивления катушек и сопротивления, описанного в данном элементе (SetNextAttributeValueAsDouble)
	2. Если не удалось записать значение
		1. Записать ошибку в массив
4. Если тег элемента «capacitor»
	1. Записать в текущую ветвь сумму уже имеющегося сопротивления конденсаторов и обратного сопротивления, описанного в данном элементе (SetNextAttributeValueAsDouble)
	2. Если не удалось записать значение
		1. Записать ошибку в массив
5. Если тег элемента «startNode»
	1. Записать в текущую ветвь индекс входного узла
	2. Если атрибут не является числом
		1. Записать ошибку в массив
6. Если тег элемента «endNode»
	1. Записать в текущую ветвь индекс выходного узла
	2. Если атрибут не является числом
		1. Записать ошибку в массив
	3. Если индекс входного узла совпадает с индексом выходного узла
		1. Вывести ошибку о повторении в одной ветви входного и выходного узлов
		2. Записать ошибку в массив
7. Если тег элемента «currentSource»
	1. Записать ЭДС цепи (SetNextAttributeValueAsDouble)
	2. Если не удалось записать значение
		1. Записать ошибку в массив
	3. Записать частоту цепи (SetNextAttributeValueAsDouble)
	4. Если не удалось записать значение
		1. Записать ошибку в массив

Функция установления следующего значения атрибута как вещественное число

bool SetNextAttributeValueAsDouble(QDomElement DomElement, double\* Number, QVector <StructError> &Error, Circuit &In);

Входные данные:

 DomElement – рассматриваемый элемент дерева

 In – описание электрической цепи

Выходные данные:

 Number – переменная для занесения в нее вещественного числа

Error – массив ошибок

 true – если удалось перевести строку в число, false – в обратном случае

Алгоритм функции:

1. Если тег элемента «currentSource»
	1. Если ЭДС не было записано ранее
		1. Запомнить атрибут «e»
	2. Иначе если частота не была записана ранее
		1. Запомнить атрибут «f»
	3. Иначе
		1. Записать ошибку в массив
		2. Вернуть false
2. Иначе
	1. Запомнить атрибут «value»
3. Записать строку в массив символов
4. Проверить строку на корректность (StringCheck)
5. Если строка корректна
	1. Записать значение атрибута в Number (sscanf)
6. Иначе
	1. Вернуть false
7. Вернуть true

Функция проверки строки с описанием вещественного числа

bool StringCheck(char\* String, QVector <StructError> &Error);

Входные данные:

 String – строка с описанием вещественного числа

Выходные данные:

Error – массив ошибок

 true если строка корректна, false – в обратном случае

Алгоритм фунции:

1. Если длина строки меньше единицы
	1. Записать ошибку в массив
	2. Вернуть false
2. Для каждого символа
	1. Если символ не цифра, «-», «E»
		1. Записать ошибку в массив
		2. Вернуть false
3. Вернуть true

Функция расчета токов электрической цепи

QVector <double> CalculationRLC(Circuit& In, int CountBranch);

Входные данные:

 In – описание электрической цепи

 CountBranch – количество ветвей

Выходные данные:

 Вектор значений токов всех ветвей

Алгоритм функции:

1. Если ветвь всего одна
	1. Рассчитать полное сопротивление, равное обратной проводимости (ConductivityBranch)
	2. Записать значение тока единственной ветви, равное ЭДС на полное сопротивление, в вектор
2. Иначе
	1. Записать систему уравнений для расчета потенциалов (SystemForNodesPotentials)
	2. Рассчитать систему уравнений потенциалов (CalculationNodesSystem)
	3. Рассчитать токи во всех ветвях (CalculationCurrentOfBranches)
3. Вернуть вектор токов цепи

Функция расчета проводимости ветви

double ConductivityBranch(int IndexBranch);

Входные данные:

IndexBranch – индекс ветви, для которой необходимо найти проводимость

Выходные данные:

Возвращает значение проводимости заданной ветви

Алгоритм функции:

1. Записать в активное сопротивление значение сопротивления резисторов данной ветви
2. Рассчитать реактивное сопротивление катушек по формуле XL=2πfL
3. Рассчитать реактивное сопротивление конденсаторов XC=$\frac{1}{2πfC}$
4. Рассчитать полное сопротивление по формуле Z=$\sqrt{R^{2}+(X\_{L}-X\_{C})^{2}}$
5. Вернуть обратное значение полного сопротивления

Функция записи системы уравнений для расчета потенциалов

void SystemForNodesPotentials (int CountNodes, Circuit& In, QVector <QVector<double>> &PotentialSystem);

Входные данные:

CountNodes – количество узлов в цепи

In – описание цепи

Выходные данные:

PotentialSystem – система уравнений для потенциалов узлов

Алгоритм функции:

1. Для каждого узла, кроме последнего
	1. Для каждого потенциала узлов, кроме последнего
		1. Приравнять элемент PotentialSystem к нулю
	2. Для каждой ветви
		1. Если в ветви данный узел является входным или выходным
			1. Найти проводимость рассматриваемой ветви с помощью функции ConductivityBranch
			2. Вычесть из коэффициента потенциала данного узла проводимость данной ветви
			3. Если второй узел из этой ветви не последний
				1. Прибавить к коэффициенту его потенциала проводимость этой ветви

Функция расчета системы уравнений для расчета потенциалов

void CalculationNodesSystem(int CountNodes, QVector <QVector<double>> &PotentialSystem, QVector <double> &Potentials);

Входные данные:

CountNodes – количество узлов в цепи

PotentialSystem – система уравнений для потенциалов узлов

Выходные данные:

Potentials – список значений потенциалов узлов

Алгоритм функции:

1. Для каждой строки системы уравнений, кроме последней
	1. Для каждой строки ниже данной
		1. Преобразовать элементы с начала до элемента на главной диагонали к нулю (TransformationEquationLine)
2. Для каждой строки системы уравнений, кроме первой
	1. Для каждой строки выше данной
		1. Преобразовать элементы с конца строки до элемента на главной диагонали к нулю (TransformationEquationLine)
3. Для каждой строки
	1. Получить значение потенциала, разделив последний элемент строки на элемент главной диагонали в этой строке

Функция приведения коэффициентов строки системы уравнений до главной диагонали к нулю

void TransformationEquationLine(QVector<QVector<double>> &PotentialSystem, int CountNodes, int FirstLine, int SecondLine)

Входные данные:

PotentialSystem – система уравнений для потенциалов узлов

CountNodes – количество узлов в цепи

FirstLine – индекс строки, относительно которой нужно преобразовывать вторую

SecondLine – индекс строки, которую нужно преобразовать

Выходные данные отсутствуют

Алгоритм функции:

1. Рассчитать отношение между второй и первой строкой по индексу элемента главной диагонали первой строки
2. Для каждого элемента строки
	1. Записать в элемент второй строки значение, равное разности этого значения и значения из первой строки с таким же индексом умноженное на отношение двух строк

Функция расчета тока в ветвях с помощью потенциалов узлов

void CalculationCurrentOfBranches(QVector<double> Potentials, QVector <double> &Currents);

Входные данные:

Potentials – список значений потенциалов узлов

Выходные данные:

Currents – список значений токов цепи

Алгоритм функции:

1. Для каждой ветви
	1. Найти проводимость рассматриваемой ветви с помощью функции ConductivityBranch
	2. Рассчитать ток ветви по формуле Ii=( φi- φ)Gi, где φi – входной узел, φ – выходной узел, Gi – проводимость ветви

Функция вывода значений тока в файл

int OutputCurrents(QFile filename, QVector<double> &Currents);

Входные данные:

Currents – список значений токов ветвей

Выходные данные:

1, если успешно, 0 – если вывести данные в файл не удалось

Алгоритм функции:

1. Открыть файл
2. Если файл можно открыть для записи
	1. Для каждой ветви
		1. Записать описание тока ветви в форме «In=v A»
3. Иначе

3.1 Вернуть false

1. Закрыть файл
2. Вернуть true

Приложение В

Диаграмма вызовов функций



Рисунок 1. Диаграмма вызовов функций

Приложение Г

Диаграмма потоков данных



Рисунок 2. Диаграмма потоков данных