**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Факультет инфокоммуникационных Сетей и систем (иксс)**

**кафедра программной инженерии и вычислительной техники**

**(пи и вт)**

|  |  |
| --- | --- |
| Направление подготовки: | 09.03.01 – Информатика и вычислительная техника |
| Кафедра: | “ССиПД” |

|  |
| --- |
| **Курсовая работа**  по дисциплине: |
| **Программирование** |

на тему:

|  |
| --- |
| **Анализ сигнала на выходе электрической цепи** |

**вариант №126**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Выполнил студент группы | ИКВТ-31 |
|  |  | Юрьева А.С | |
|  |  | *Фамилия И.О* | |
|  |  | Руководитель | Ассистент |
| оценка |  |  | уч. степень, уч. звание |
|  |  | Дятлов Д.А | |
| ***дата, подпись*** |  | ***Фамилия И.О*** | |

Санкт-Петербург

2024г

Оглавление

[Задание к курсовой работе 2](#_Toc164779335)

[Таблица идентификаторов 3](#_Toc164779336)

[Блок-схемы программы 4](#_Toc164779337)

[Графический интерфейс программы 4](#_Toc164779338)

[Текст программы 4](#_Toc164779339)

[Course.cpp 4](#_Toc164779340)

[Course.h 6](#_Toc164779341)

[Func.cpp 13](#_Toc164779342)

[Func.h 14](#_Toc164779343)

[Вывод 14](#_Toc164779344)

[Список используемой литературы 14](#_Toc164779345)

[Приложения к пояснительной записке 14](#_Toc164779346)

# Задание к курсовой работе

Работа посвящена решению задач машинного анализа электрических цепей.

В курсовой работе необходимо для заданной электрической цепи по известному входному сигналу определить выходной сигнал для N равностоящих моментов времени, а затем определить некоторые его характеристики с погрешностью не более 1%.

Вариант №126

Параметры входного сигнала электрической цепи:

Параметры передаточной характеристики электрической цепи:

В ходе выполнения работы:

* Произвести расчет входного и выходного сигнала в контрольных точках, используя при этом математический пакет Wxmaxima;
* Построить графики зависимостей входного и выходного напряжений от времени, используя математический пакет Wxmaxima;
* Произвести расчет входного и выходного сигнала в контрольных точках при помощи языка программирования C++;
* Сравнить полученные результаты, оформить контрольный расчет;
* Произвести расчет параметра на контрольных точках вручную и при помощи языка программирования С++;
* Написать алгоритм вычисления длительности импульса выходного сигнала напряжения;
* Написать алгоритм вычисления длительности импульса выходного сигнала напряжения с заданной погрешностью при помощи языка программирования C++;
* Оформить графический интерфейс при помощи Winforms CLI;
* Построить графики на графическом интерфейсе при помощи компонента chart;
* Организовать вывод значений функций зависимостей входного и выходного напряжения от времени, значений параметра с заданной погрешностью на графический интерфейс;
* Произвести отладку программы: оформить табулированный вывод, обработку вводимых значений при помощи операторов обработки исключений try catch;
* Оформить пояснительную записку, используя текстовый редактор MS Word;
* Защитить работу преподавателю.

# Таблица идентификаторов

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Идентификатор | Значение | Ед. измерения |
| 1 | a | 12 | - |
| 2 | b | 12 | - |
| 3 | tнач | 10 | Сек |
| 4 | tкон | 35 | Сек |
| 5 | t1 | 22,5 | Сек |
| 6 | U1 | 20 | В |
| 7 | U2 | 150 | В |
| 8 | U3 | 20 | В |
| 9 | U4 | 100 | В |

# Блок-схема функции main()

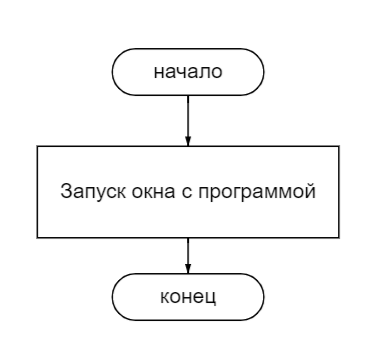


Рис.1 Блок-схема функции main()

# Графический интерфейс программы

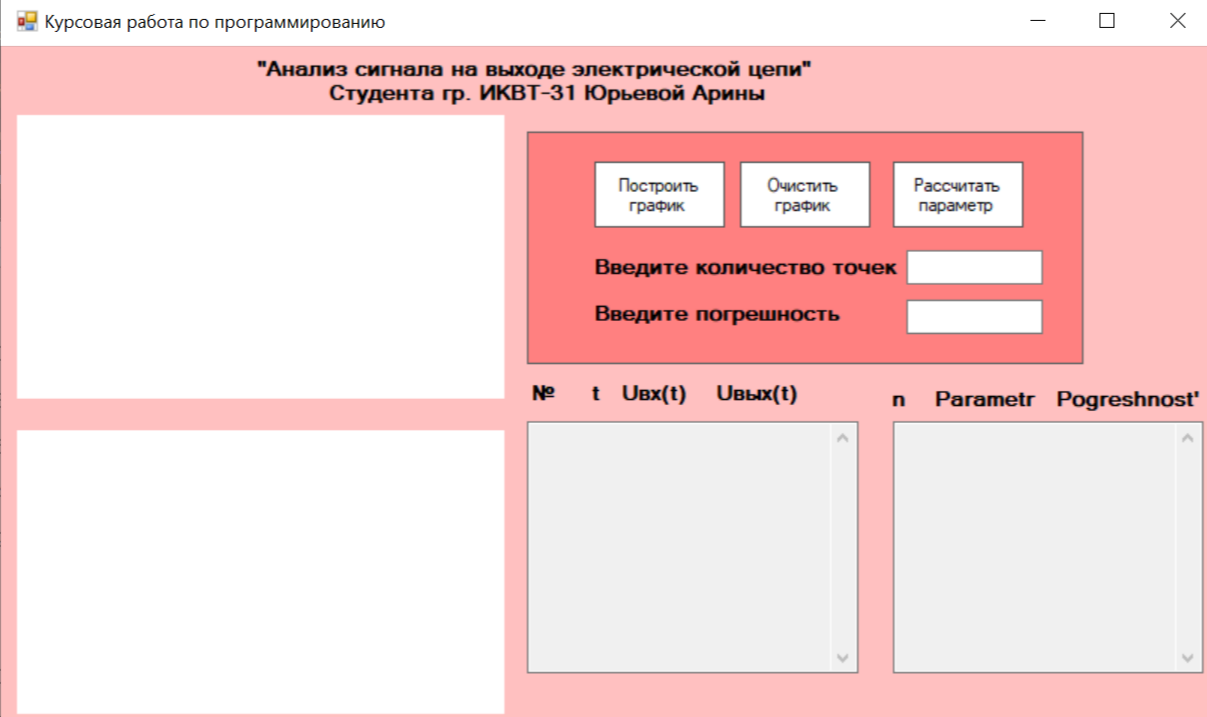


Рис.2 Графический интерфейс в изначальном виде

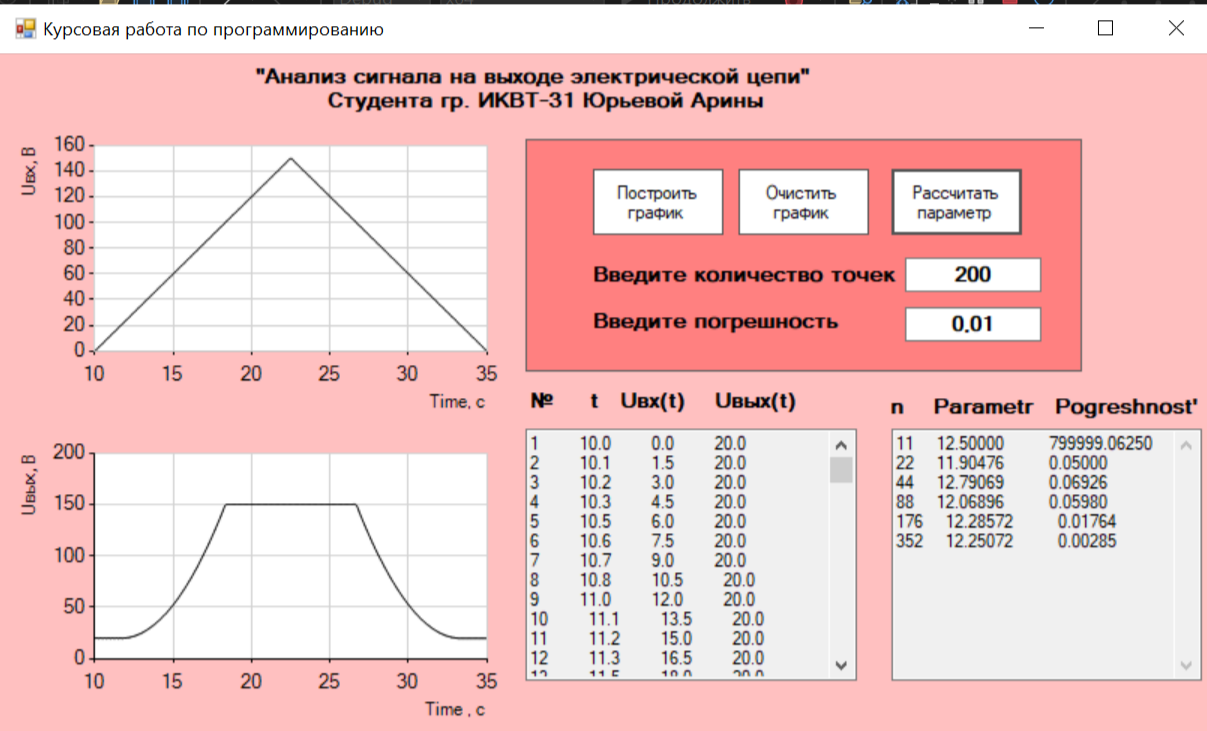


Рис.3 Графический интерфейс и функционал программы

# Текст программы

Course.cpp

#include "App.h"

#include "func.h"

using namespace System;

using namespace System::Windows::Forms;

using namespace System::Globalization;

[STAThreadAttribute]

int main(array<String^>^ args) {

Application::SetCompatibleTextRenderingDefault(false);

Application::EnableVisualStyles();

coursepaper::App window;

Application:: Run(% window);

}

System::Void coursepaper::App::button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->chart->Series[0]->Points->Clear();

this->chart1->Series[0]->Points->Clear();

try { this->num\_dots = Int32::Parse(textBox1->Text); }

catch (...) {};

if (this->num\_dots > 1) {

int len = Int32::Parse(textBox1->Text);

float\* time = new float[len];

float\* Uin = new float[len];

float\* Uout = new float[len];

arr\_time(time, len);

arr\_Uin(Uin, time, len);

arr\_Uout(Uout, Uin, len);

for (int i = 0; i < len; i++) {

std::string space = " ";

this->chart->Series[0]->Points->AddXY(time[i], Uout[i]);

this->chart1->Series[0]->Points->AddXY(time[i], Uin[i]);

std::string value =std::to\_string(i + 1) + space + el\_1(std::to\_string(time[i]), 1)

+space+ el\_1(std::to\_string(Uin[i]), 1) +space+ el\_1(std::to\_string(Uout[i]), 1);

String^ new\_line = "\r\n";

String^ textValue = gcnew String(value.c\_str());

textBox2->Text += textValue;

textBox2->Text += new\_line;

}

this->chart->ChartAreas[0]->AxisX->Minimum = time[0];

this->chart->ChartAreas[0]->AxisX->Maximum = time[len-1];

this->chart1->ChartAreas[0]->AxisX->Minimum = time[0];

this->chart1->ChartAreas[0]->AxisX->Maximum = time[len - 1];

delete[] time;

delete[] Uin;

delete[] Uout;

}

else {

this->chart->Series[0]->Points->Clear();

this->chart1->Series[0]->Points->Clear();

this->textBox2->Clear();

}

return System::Void();

}

System::Void coursepaper::App::button2\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->chart->Series[0]->Points->Clear();

this->chart1->Series[0]->Points->Clear();

this->textBox2->Clear();

this->textBox3->Clear();

return System::Void();

}

System::Void coursepaper::App::textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

return System::Void();

}

System::Void coursepaper::App::button3\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->textBox3->Clear();

float eps;

try { eps = (float)Double::Parse(textBox4->Text); }

catch (...) {};

if (eps != 0) {

float p = 1; //текущая погрешность

float par = 10000000; //текущее значение параметра

int len = 11;

while (p > eps) {

float\* time = new float[len];

float\* Uin = new float[len];

float\* Uout = new float[len];

arr\_time(time, len);

arr\_Uin(Uin, time, len);

arr\_Uout(Uout, Uin, len);

float par\_1 = parametr(Uout, len);

p = fabs(par - par\_1) / par\_1;

std::string space = " ";

std::string value = std::to\_string(len) + space + el\_1(std::to\_string(par\_1), 5) + space + space +

el\_1(std::to\_string(p), 5);

String^ new\_line = "\r\n";

String^ textValue = gcnew String(value.c\_str());

textBox3->Text += textValue;

textBox3->Text += new\_line;

par = par\_1;

len \*= 2;

delete[] time;

delete[] Uin;

delete[] Uout;

}

}

return System::Void();

}

Course.h

#pragma once

#include "func.h"

namespace coursepaper {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class App : public System::Windows::Forms::Form

{

public:

App(void)

{

InitializeComponent();

}

protected:

~App()

{

if (components)

{

delete components;

}

}

private:

int num\_dots;

String^ cur\_line;

System::Windows::Forms::DataVisualization::Charting::Chart^ chart;

protected:

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::Label^ label6;

private:

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea2 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Series^ series2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

this->chart = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->chart1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->button3 = (gcnew System::Windows::Forms::Button());

this->label2 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->label6 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->BeginInit();

this->SuspendLayout();

//

// chart

//

this->chart->AllowDrop = true;

this->chart->BackColor = System::Drawing::Color::Transparent;

this->chart->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->chart->BackSecondaryColor = System::Drawing::Color::White;

this->chart->BorderlineColor = System::Drawing::Color::Black;

chartArea1->AxisX->MajorGrid->LineColor = System::Drawing::Color::LightGray;

chartArea1->AxisX->ScaleView->MinSizeType = System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Seconds;

chartArea1->AxisX->ScaleView->SizeType = System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Seconds;

chartArea1->AxisY->MajorGrid->LineColor = System::Drawing::Color::LightGray;

chartArea1->Name = L"ChartArea1";

this->chart->ChartAreas->Add(chartArea1);

this->chart->ImeMode = System::Windows::Forms::ImeMode::NoControl;

this->chart->Location = System::Drawing::Point(2, 385);

this->chart->Name = L"chart";

this->chart->Palette = System::Windows::Forms::DataVisualization::Charting::ChartColorPalette::None;

this->chart->PaletteCustomColors = gcnew cli::array< System::Drawing::Color >(1) { System::Drawing::Color::Black };

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;

series1->LabelForeColor = System::Drawing::Color::Gray;

series1->Legend = L"Legend1";

series1->Name = L"Series1";

this->chart->Series->Add(series1);

this->chart->Size = System::Drawing::Size(519, 310);

this->chart->TabIndex = 0;

//

// button1

//

this->button1->BackColor = System::Drawing::Color::White;

this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->button1->Location = System::Drawing::Point(594, 118);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(130, 68);

this->button1->TabIndex = 1;

this->button1->Text = L"Построить график";

this->button1->UseVisualStyleBackColor = false;

this->button1->Click += gcnew System::EventHandler(this, &App::button1\_Click);

//

// button2

//

this->button2->BackColor = System::Drawing::Color::White;

this->button2->BackgroundImageLayout = System::Windows::Forms::ImageLayout::None;

this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->button2->Location = System::Drawing::Point(740, 118);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(130, 68);

this->button2->TabIndex = 2;

this->button2->Text = L"Очистить график";

this->button2->UseVisualStyleBackColor = false;

this->button2->Click += gcnew System::EventHandler(this, &App::button2\_Click);

//

// textBox1

//

this->textBox1->Cursor = System::Windows::Forms::Cursors::Default;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(906, 209);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(135, 30);

this->textBox1->TabIndex = 3;

this->textBox1->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->textBox1->TextChanged += gcnew System::EventHandler(this, &App::textBox1\_TextChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(589, 212);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(289, 25);

this->label1->TabIndex = 4;

this->label1->Text = L"Введите количество точек";

//

// chart1

//

this->chart1->AllowDrop = true;

this->chart1->BackColor = System::Drawing::Color::Transparent;

this->chart1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->chart1->BackSecondaryColor = System::Drawing::Color::White;

this->chart1->BorderlineColor = System::Drawing::Color::Black;

chartArea2->AxisX->LineColor = System::Drawing::Color::Silver;

chartArea2->AxisX->MajorGrid->LineColor = System::Drawing::Color::LightGray;

chartArea2->AxisX->ScaleBreakStyle->LineColor = System::Drawing::Color::Silver;

chartArea2->AxisX->ScaleView->MinSizeType = System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Seconds;

chartArea2->AxisX->ScaleView->SizeType = System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Seconds;

chartArea2->AxisX2->LineColor = System::Drawing::Color::Silver;

chartArea2->AxisY->LineColor = System::Drawing::Color::LightGray;

chartArea2->AxisY->MajorGrid->LineColor = System::Drawing::Color::LightGray;

chartArea2->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea2);

this->chart1->ImeMode = System::Windows::Forms::ImeMode::NoControl;

this->chart1->Location = System::Drawing::Point(2, 62);

this->chart1->Name = L"chart1";

this->chart1->Palette = System::Windows::Forms::DataVisualization::Charting::ChartColorPalette::None;

this->chart1->PaletteCustomColors = gcnew cli::array< System::Drawing::Color >(1) { System::Drawing::Color::Black };

series2->ChartArea = L"ChartArea1";

series2->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;

series2->LabelForeColor = System::Drawing::Color::Gray;

series2->Legend = L"Legend1";

series2->Name = L"Series1";

this->chart1->Series->Add(series2);

this->chart1->Size = System::Drawing::Size(519, 310);

this->chart1->TabIndex = 5;

//

// button3

//

this->button3->BackColor = System::Drawing::Color::White;

this->button3->BackgroundImageLayout = System::Windows::Forms::ImageLayout::None;

this->button3->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->button3->Location = System::Drawing::Point(892, 118);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(130, 68);

this->button3->TabIndex = 6;

this->button3->Text = L"Рассчитать параметр";

this->button3->UseVisualStyleBackColor = false;

this->button3->Click += gcnew System::EventHandler(this, &App::button3\_Click);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(527, 341);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(243, 25);

this->label2->TabIndex = 7;

this->label2->Text = L"№ t Uвх(t) Uвых(t)";

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(527, 385);

this->textBox2->Multiline = true;

this->textBox2->Name = L"textBox2";

this->textBox2->ReadOnly = true;

this->textBox2->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox2->Size = System::Drawing::Size(329, 257);

this->textBox2->TabIndex = 8;

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(245, 9);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(518, 25);

this->label4->TabIndex = 11;

this->label4->Text = L" \"Анализ сигнала на выходе электрической цепи\"";

this->label4->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(324, 34);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(408, 25);

this->label5->TabIndex = 12;

this->label5->Text = L"Студента гр. ИКВТ-31 Юрьевой Арины ";

this->label5->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label3

//

this->label3->AutoSize = true;

this->label3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(589, 260);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(237, 25);

this->label3->TabIndex = 14;

this->label3->Text = L"Введите погрешность";

//

// textBox4

//

this->textBox4->Cursor = System::Windows::Forms::Cursors::Default;

this->textBox4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox4->Location = System::Drawing::Point(906, 260);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(135, 30);

this->textBox4->TabIndex = 13;

this->textBox4->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(892, 385);

this->textBox3->Multiline = true;

this->textBox3->Name = L"textBox3";

this->textBox3->ReadOnly = true;

this->textBox3->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox3->Size = System::Drawing::Size(308, 257);

this->textBox3->TabIndex = 15;

//

// textBox5

//

this->textBox5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->textBox5->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox5->ForeColor = System::Drawing::SystemColors::AppWorkspace;

this->textBox5->Location = System::Drawing::Point(527, 87);

this->textBox5->Multiline = true;

this->textBox5->Name = L"textBox5";

this->textBox5->ReadOnly = true;

this->textBox5->Size = System::Drawing::Size(555, 237);

this->textBox5->TabIndex = 16;

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label6->Location = System::Drawing::Point(887, 347);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(279, 25);

this->label6->TabIndex = 17;

this->label6->Text = L"n Parametr Pogreshnost\'";

//

// App

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(192)),

static\_cast<System::Int32>(static\_cast<System::Byte>(192)));

this->ClientSize = System::Drawing::Size(1212, 940);

this->Controls->Add(this->label6);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->label3);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->button3);

this->Controls->Add(this->chart1);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->chart);

this->Controls->Add(this->textBox5);

this->ForeColor = System::Drawing::SystemColors::MenuText;

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;

this->Name = L"App";

this->Text = L"Курсовая работа по программированию";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e);

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e);

}

; }

Func.cpp

#include "func.h"

using namespace std;

int valid\_enter(int\* n) {

try {

string \_n;

cin >> \_n;

\*n = stoi(\_n);

}

catch (...) {

cout << "Error! Wrong data type entered\n";

return 0;

}

return 1;

}

void arr\_time(float\* time, int len) {

float dt = (t\_fin - t\_st) / float(len - 1);

for (int i = 0; i < len; i++) {

time[i] = t\_st + i \* dt;

}

}

void arr\_Uin(float\* Uin, float\* time, int len) {

for (int i = 0; i < len; i++) {

if (time[i] <= t1) Uin[i] = a \* (time[i] - t\_st);

else Uin[i] = a \* (t1 - t\_st) - b \* (time[i] - t1);

}

}

void arr\_Uout(float\* Uout, float\* Uin, int len) {

for (int i = 0; i < len; i++) {

if (Uin[i] <= U3) Uout[i] = U1;

else if (Uin[i] <= U4) Uout[i] = U1 + (U2 - U1) \* pow((Uin[i] - U3), 2) / pow((U4 - U3), 2);

else Uout[i] = U2;

}

}

std::string el\_1(std::string el\_, int len)

{

std::string el;

int index = el\_.find('.');

int i = 0;

while (i <= index + len) {

el += el\_[i];

i++;

}

return std::string(el);

}

float parametr(float\* Uout, int len) {

float U\_max = 0, U\_min = 10000;

for (int i = 0; i < len; i++) {

if (Uout[i] < U\_min) U\_min = Uout[i];

if (Uout[i] > U\_max) U\_max = Uout[i];

}

float U\_imp = U\_min + 0.5 \* (U\_max - U\_min);

float dlit = 0, dt = 25. / (len - 1);

for (int i = 0; i < len; i++) {

if (Uout[i] >= U\_imp) dlit += dt;

}

return dlit;

}

Func.h

#include <iostream>

# include <stdio.h>

# include <string>

# include <math.h>

# include <vector>

#define a 12

#define b 12

#define t\_st 10

#define t\_fin 35

#define t1 22.5

#define U1 20

#define U2 150

#define U3 20

#define U4 100

void arr\_time(float\* time, int len);

void arr\_Uin(float\* Uin, float\* time, int len);

void arr\_Uout(float\* Uout, float\* Uin, int len);

std::string el\_1(std::string el\_, int len);

float parametr(float\* Uout, int len);

# Вывод

В ходе выполнения курсовой работы были получены практические навыки создания приложений с графическим интерфейсом при помощи Winforms C++/CLI и использования математического пакета Wxmaxima для реализации расчетов и построения графиков. Кроме того, были изучены методы и компоненты среды Winforms CLR и технология разработки приложения с графическим интерфейсом.

Было разработано и реализовано приложение с графическим интерфейсом пользователя на основе Winforms CLI, которое осуществляет вычисление значений функций зависимости входного и выходного напряжения от времени, а также строит графики данных зависимостей. Кроме того, программа вычисляет длительность импульса выходного напряжения с заданной погрешностью и рассчитывает наиболее оптимальное количества точек для достижения желаемой точности.

# Список используемой литературы

1. Методические материалы “Информатика. Анализ сигнала на выходе электрической цепи” Л.Н. Бережной И.О. Воронцова Д.В. Окунева

# Приложения к пояснительной записке

1. Контрольный расчет