**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,  
СВЯЗИ И МАССОВЫХ ТЕЛЕКОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет «Инфокоммуникационных сетей и систем»

Кафедра «Системы связи и передачи данных»

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

**КУРСОВАЯ РАБОТА**

по дисциплине:

|  |
| --- |
| **ПРОГРАММИРОВАНИЕ** |

на тему:

|  |
| --- |
| **АНАЛИЗ СИГНАЛА НА ВЫХОДЕ ЭЛЕКТИЧЕСКОЙ ЦЕПИ** |

Вариант задания – 128

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Выполнила студентка группы | | ИКВТ-42 |
|  |  | Senda Ouahchi | | |
|  |  | Фамилия И. О. | | |
|  |  | Руководитель | ассистент | |
| оценка |  |  | уч. степень, уч. звание | |
|  |  | Дятлов Д. А. | | |
| дата, подпись |  | Фамилия И. О. | | |

## Аннотация

Разработанное программное обеспечение предназначено для моделирования и анализа характеристик выходных сигналов электрических цепей в соответствии с методическими указаниями СПбГУТ. Приложение реализует численный расчет выходного напряжения на основе заданной математической модели входного сигнала и передаточной функции цепи.

**Технические особенности реализации:**

* Язык программирования: C++
* Использование стандартных математических библиотек
* Обеспечение высокой точности вычислений (double precision)

**Функциональные возможности:**

1. Расчет амплитудно-временных характеристик сигнала:
   * Формирование массивов временных отсчетов
   * Расчет значений входного и выходного напряжений
   * Определение параметров сигнала (длительность фронтов, амплитуда)
2. Дополнительные функции:
   * Экспорт результатов в текстовые файлы
   * Визуализация данных (совместимость с Wxmaxima и OpenOffice Calc)
   * Интерактивное меню управления
3. Контроль точности:
   * Автоматический расчет с адаптивным шагом
   * Обеспечение заданной погрешности (не более 1%)

Программа прошла тестирование, подтвердившее точность моделирования и соответствие ожидаемым результатам.

## Оглавление

[Аннотация 2](#_Toc199795468)

[Оглавление 3](#_Toc199795469)

[Задание 4](#_Toc199795470)

[Таблица идентификаторов 5](#_Toc199795471)

[Блок схема 9](#_Toc199795472)

[Текст программы 10](#_Toc199795473)

[Код на cpp 10](#_Toc199795474)

[wxMaxima код 19](#_Toc199795475)

[Описание программы 21](#_Toc199795476)

[Общие сведения 21](#_Toc199795477)

[Исполняемые файлы: 21](#_Toc199795478)

[Функциональное назначение: 21](#_Toc199795479)

[Алгоритм работы: 21](#_Toc199795480)

[Технические требования: 22](#_Toc199795481)

[Графики 23](#_Toc199795482)

[Интерфейс 24](#_Toc199795483)

[Заключение 25](#_Toc199795484)

[Список литературы 26](#_Toc199795485)

[Приложение 27](#_Toc199795486)

## Задание

В курсовой работе требуется:

1. Для заданной электрической цепи:
   * Определить выходной сигнал для N равностоящих моментов времени.
   * Рассчитать характеристики сигнала с погрешностью не более 1%.
2. Этапы выполнения:
   * Контрольный расчёт в Wxmaxima.
   * Разработка программы на C++ (среда NetBeans).
   * Запись результатов в файлы и построение графиков (Wxmaxima или OpenOffice Calc).
   * Оформление пояснительной записки по требованиям методички.
   * Защита работы перед преподавателем.

## Таблица идентификаторов

error\_struct.h

| **Название** | **Тип** | **Предназначение** |
| --- | --- | --- |
| Error | struct | Хранит данные о расчёте параметров и погрешности |
| N | int | Количество точек расчёта |
| Uvx\_parametr | double | Значение параметра входного сигнала |
| Uvix\_parametr | double | Значение параметра выходного сигнала |
| error\_rate | double | Погрешность расчёта |

io.cpp

| **Название** | **Тип** | **Предназначение** |
| --- | --- | --- |
| print\_arrays | void function | Выводит таблицу значений времени, Uvx, Uvix |
| save\_array | void function | Сохраняет массив данных в файл |
| print\_struct | void function | Выводит таблицу с результатами и погрешностями |
| readFile | void function | Читает и выводит содержимое файла |
| input\_number\_points | int function | Запрашивает у пользователя количество точек |
| dir | filesystem::path | Путь для сохранения файлов |
| file | ofstream/ifstream | Поток для работы с файлами |

logic.cpp

| **Название** | **Тип** | **Предназначение** |
| --- | --- | --- |
| U1, U2, T | const double | Параметры входного сигнала (амплитуда, период) |
| tn, tk | const double | Начальное и конечное время |
| V1, V2 | const double | Граничные значения выходного сигнала |
| Uvx1, Uvx2 | const double | Пороговые значения входного напряжения |
| Uvx(t) | double function | Вычисляет входное напряжение в момент t |
| Uvix(t) | double function | Вычисляет выходное напряжение в момент t |
| get\_io\_voltage | void function | Заполняет массив Uvx или Uvix |
| get\_time | void function | Формирует массив времени |
| fill\_all\_arrays | void function | Заполняет все массивы (время, Uvx, Uvix) |
| get\_impuls\_length | double function | Вычисляет длительность импульса сигнала |
| get\_param\_with\_an\_error | vector<Error> function | Вычисляет параметры с заданной точностью |

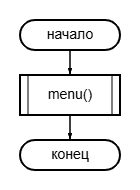
menu.cpp

| **Название** | **Тип** | **Предназначение** |
| --- | --- | --- |
| CLEAR\_SCREEN() | macro | Очищает консоль (зависит от ОС) |
| print\_parameter | void function | Выводит длительность импульса Uvx и Uvix |
| print\_write\_voltages | void function | Выводит или сохраняет массивы данных |
| menu | void function | Основное меню программы |
| choice | int | Выбор пользователя в меню |
| stop | string | Переменная для подтверждения выхода |

main.cpp

| **Название** | **Тип** | **Предназначение** |
| --- | --- | --- |
| main | int function | Точка входа в программу, вызывает menu() |

## Блок схема



1. Функция main

## Текст программы

### Код на cpp

##### main.cpp

#include "menu.h"

int main() {

menu();

return 0;

}

##### menu.cpp

#include "menu.h"

#include "io.h"

#include "logic.h"

#include <iostream>

#include <cstdlib>

#include <iomanip>

using namespace std;

#ifdef \_WIN32

#define CLEAR\_SCREEN() system("cls")

#else

#define CLEAR\_SCREEN() system("clear")

#endif

void print\_parameter() {

int N = input\_number\_points();

vector<double> Uvx\_array(N), Uvix\_array(N), time\_array(N);

fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

cout << "Input signal pulse width: " << setprecision(7) << get\_impuls\_length(N, Uvx\_array)

<< "\nOutput signal pulse width: " << get\_impuls\_length(N, Uvix\_array);

}

void print\_write\_voltages(bool write) {

int N = input\_number\_points();

vector<double> Uvx\_array(N), Uvix\_array(N), time\_array(N);

fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

if (!write) {

print\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

}

else {

save\_array(N, time\_array, "data/time\_array.txt");

save\_array(N, Uvx\_array, "data/Uvx\_array.txt");

save\_array(N, Uvix\_array, "data/Uvix\_array.txt");

cout << "Done";

}

}

void menu() {

int choice;

char stop;

while (true) {

readFile("zastavka.txt");

cout << "1 - Control calculation for N points\n";

cout << "2 - Calculation of a parameter with N points\n";

cout << "3 - Calculation of a parameter with a given accuracy\n";

cout << "4 - Writing data to a file\n";

cout << "5 - Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch(choice) {

case 1:

print\_write\_voltages(false);

break;

case 2:

print\_parameter();

break;

case 3:

print\_struct(get\_param\_with\_an\_error());

break;

case 4:

print\_write\_voltages(true);

break;

case 5:

return;

default:

CLEAR\_SCREEN();

continue;

}

cout << "\nTerminate?[Y/N] ";

cin.ignore();

cin.get(stop);

if (stop == 'y' || stop == 'Y') {

return;

}

CLEAR\_SCREEN();

}

}

##### io.cpp

#include "io.h"

#include "error\_struct.h"

#include <iostream>

#include <fstream>

#include <iomanip>

#include <filesystem>

#include <windows.h>

using namespace std;

void print\_arrays(int N, const vector<double>& time, const vector<double>& Uvx, const vector<double>& Uvix) {

cout << "+------+------------+----------+----------+\n";

cout << "| No | time | Uvx | Uvix |\n";

cout << "+------+------------+----------+----------+\n";

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

cout << "| " << setw(4) << i + 1

<< " | " << setw(10) << fixed << setprecision(7) << time[i]

<< " | " << setw(8) << setprecision(4) << Uvx[i]

<< " | " << setw(8) << Uvix[i] << " |\n";

}

cout << "+------+------------+----------+----------+\n";

}

void save\_array(int N, const vector<double>& array\_, const string& name) {

filesystem::path dir = filesystem::path(name).parent\_path();

if (!dir.empty() && !filesystem::exists(dir)) {

filesystem::create\_directories(dir);

}

ofstream file(name);

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

cerr << "File error: " << name << endl;

return;

}

file << fixed << setprecision(5);

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

file << array\_[i] << "\n";

}

file.close();

}

void print\_struct(const vector<Error> error\_list){

cout << "+-----+--------------|---------------+------------+\n";

cout << "| N | Input length | Output length | Error rate |\n";

cout << "+-----+--------------|---------------+------------+\n";

for (int i = 0; i < error\_list.size(); i++){

cout << "| " << setw(3) << error\_list[i].N

<< " | " << setw(12) << fixed << setprecision(6) << error\_list[i].Uvx\_parametr

<< " | " << setw(13) << fixed << setprecision(6) << error\_list[i].Uvix\_parametr

<< " | " << setw(10) << (error\_list[i].error\_rate > 9999 ? scientific : fixed)

<< setprecision(4) << error\_list[i].error\_rate << " |\n";

}

cout << "+-----+--------------|---------------+------------+\n";

}

void readFile(const string& name) {

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

ifstream file(name);

string line;

while (getline(file, line)) {

cout << line << endl;

}

}

int input\_number\_points() {

int N;

cout << "Enter number of points: ";

cin >> N;

return N;

}

##### logic.cpp

#include "logic.h"

#include "error\_struct.h"

#include <cmath>

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

// constanti dlya KOD A

constexpr double U = 250.0;

// NACHALNAYA i KONECHNAYA TOCHKI

constexpr double tn = -10.0;

constexpr double tk = 10.0;

// constanti dlya KOD B

constexpr double a = 400.0;

constexpr double U0 = 50.0;

constexpr double U1 = 100.0;

double Uvx(double t)

{

if (t <= 0)

{

return (t - tn) \* (U - 0) / (0 - tn) + 0;

} else

{

return (t - 0) \* (0 - U) / (tk - 0) + U;

}

}

double Uvix(double t)

{

if (0 <= Uvx(t) && Uvx(t) <= U0)

{

return pow(a \* pow(Uvx(t), 2), (1.0 / 3.0));

} else if (Uvx(t) > U0)

{

return U1;

}

}

void get\_io\_voltage(int N, const vector<double>& time\_array, vector<double>& output\_array, int func) {

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

output\_array[i] = (func == 0) ? Uvx(time\_array[i]) : Uvix(time\_array[i]);

}

}

void get\_time(int N, vector<double>& output\_array) {

double dt = (tk - tn) / (N - 1.0);

double t = tn;

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

output\_array[i] = t;

t += dt;

}

}

void fill\_all\_arrays(int N, vector<double>& time\_array, vector<double>& Uvx\_array, vector<double>& Uvix\_array) {

get\_time(N, time\_array);

get\_io\_voltage(N, time\_array, Uvx\_array, 0);

get\_io\_voltage(N, time\_array, Uvix\_array, 1);

}

double get\_impuls\_length(int N, const vector<double>& input\_array) {

double max = input\_array[0];

double min = input\_array[0];

for (int i = 1; i < N; i++) {

if (input\_array[i] > max) max = input\_array[i];

if (input\_array[i] < min) min = input\_array[i];

}

double dt = (tk - tn) / (N - 1.0);

double Uavg = (max + min) / 2.0;

double dlit = 0;

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

if (input\_array[i] > Uavg) {

dlit += dt;

}

}

return dlit;

}

vector<Error> get\_param\_with\_an\_error() {

double error\_rate\_Uvx = 1, error\_rate\_Uvix = 1;

constexpr double eps = 0.01;

double par\_Uvx = 1e+10, par1\_Uvx;

double par\_Uvix = 1e+10, par1\_Uvix;

int N = 11, c = 0;

vector<Error> error\_list;

while (error\_rate\_Uvx > eps && error\_rate\_Uvix > eps) {

vector<double> Uvx\_array(N), Uvix\_array(N), time\_array(N);

fill\_all\_arrays(N, time\_array, Uvx\_array, Uvix\_array);

par1\_Uvx = get\_impuls\_length(N, Uvx\_array);

par1\_Uvix = get\_impuls\_length(N, Uvix\_array);

error\_rate\_Uvx = fabs(par\_Uvx - par1\_Uvx) / par1\_Uvx;

error\_rate\_Uvix = fabs(par\_Uvix - par1\_Uvix) / par1\_Uvix;

Error error {N, par1\_Uvx, par1\_Uvix, max(error\_rate\_Uvx, error\_rate\_Uvix)};

error\_list.push\_back(error);

par\_Uvx = par1\_Uvx;

par\_Uvix = par1\_Uvix;

N \*= 2;

c++;

}

return error\_list;

}

##### menu.h

#ifndef MENU\_H\_INCLUDED

#define MENU\_H\_INCLUDED

void menu();

#endif // MENU\_H\_INCLUDED

##### io.h

#ifndef IO\_H\_INCLUDED

#define IO\_H\_INCLUDED

#include <vector>

#include <string>

#include "error\_struct.h"

void print\_arrays(int N, const std::vector<double>& time, const std::vector<double>& Uvx, const std::vector<double>& Uvix);

void save\_array(int N, const std::vector<double>& array, const std::string& name);

int input\_number\_points();

void readFile(const std::string& name);

void print\_struct(const std::vector<Error> error\_list);

#endif // IO\_H\_INCLUDED

##### logic.h

#ifndef LOGIC\_H\_INCLUDED

#define LOGIC\_H\_INCLUDED

#include <vector>

#include "error\_struct.h"

double Uvx(double t);

double Uvix(double t);

void get\_io\_voltage(int N, const std::vector<double>& time\_array, std::vector<double>& output\_array, int func);

void get\_time(int N, std::vector<double>& output\_array);

void fill\_all\_arrays(int N, std::vector<double>& time\_array, std::vector<double>& Uvx\_array, std::vector<double>& Uvix\_array);

double get\_impuls\_length(int N, const std::vector<double>& input\_array);

std::vector<Error> get\_param\_with\_an\_error();

#endif // LOGIC\_H\_INCLUDED

##### error\_struct.h

#ifndef ERROR\_STRUCT\_H\_INCLUDED

#define ERROR\_STRUCT\_H\_INCLUDED

struct Error{

int N;

double Uvx\_parametr;

double Uvix\_parametr;

double error\_rate;

};

#endif // ERROR\_STRUCT\_H\_INCLUDED

### wxMaxima код

fpprintprec:5$;

U: 5$; T: 2 \* 10^-5$;

tn: 0$;

tk: 2 \* T$;

U1: 2$; U2: 50$;

N: 50$; dt: (tk - tn) / (N - 1)$;

Uvx(t) :=

if (0 <= t and t < T / 2) then

2 \* U \* t / T

else if (T / 2 <= t and t < T) then

2 \* U \* (T-t) / T

else if (T <= t and t < 3 \* T / 2) then

2 \* U \* (t - T) / T

else if (3 \* T / 2 <= t and t <= 2 \* T) then

2 \* U \* (2 \* T - t) / T

$;

Uvix(t):=

if Uvx(t) < U1 then

U2 / U1^2 \* Uvx(t) \* (2 \* U1 - Uvx(t))

else

U2

$;

for i: 0 thru N-1 do (

t: float(tn + i \* dt),

display(Uvx(t), Uvix(t))), numer;

/\*

data\_t: read\_list("data/time\_array.txt");

data\_Uvx: read\_list("data/Uvx\_array.txt");

data\_Uvix: read\_list("data/Uvix\_array.txt");

wxplot2d([[discrete, data\_t, data\_Uvx]],

[gnuplot\_preamble, "set grid;"],

[xlabel, ""],

[ylabel, ""],

[title, "График функции Uvx(t) средствами C++"])$;

\*/

wxplot2d([Uvx(time)], [time, tn, tk],

[gnuplot\_preamble, "set grid;"],

[xlabel, ""],

[ylabel, ""],

[title, "График функции Uvx(t) средствами wxMaxima"])$;

/\*

wxplot2d([[discrete, data\_t, data\_Uvix]],

[gnuplot\_preamble, "set grid;"],

[xlabel, ""],

[ylabel, ""],

[title, "График функции Uvix(t) средствами C++"])$;

\*/

wxplot2d([Uvix(time)], [time, tn, tk],

[gnuplot\_preamble, "set grid;"],

[xlabel, ""],

[ylabel, ""],

[title, "График функции Uvix(t) средствами wxMaxima"])$;

## Описание программы

### Общие сведения

Программа анализирует сигналы в электрической цепи, вычисляет выходное напряжение и сохраняет результаты для визуализации.

### ***Исполняемые файлы***:

* *cpp.exe – основная программа на C++.*
* *kpm.wxmx – файл расчётов в Wxmaxima.*

### Функциональное назначение:

1. *Расчёт сигналов:*
   * Формирование массивов времени, входного (Uvx) и выходного (Uvix) напряжений.
   * Определение параметров сигнала (например, длительности импульса).
2. *Сохранение данных:*
   * Результаты записываются в файлы massiv\_t.txt, massiv\_Uvx.txt, massiv\_Uvix.txt.
3. *Визуализация:*
   * Построение графиков в Wxmaxima.

### Алгоритм работы:

1. Пользователь запускает программу, выбирает режим через меню.
2. Программа выполняет расчёты и сохраняет данные.
3. Графики строятся автоматически на основе сохранённых файлов.

### Технические требования:

* Среда разработки: CodeBlocks (C++).
* Математический пакет: Wxmaxima.
* Дополнительно: OpenOffice Calc для альтернативной визуализации.

## Графики

|  |  |  |
| --- | --- | --- |
|  | cpp | wxMaxima |
| Uvx(t) |  |  |
| Uvix(t) |  |  |

На основании проведённого анализа сигналов и построенных графиков можно сделать следующие выводы:

**Совпадение результатов расчётов**  
Результаты, полученные при ручном расчёте в среде WxMaxima, полностью совпадают с данными, вычисленными программой на C++. Это подтверждает:

* + Корректность аналитических формул для Uвх(t) и Uвых​(t).
  + Правильность алгоритмов, реализованных в программе.
  + Точность численных методов (погрешность ≤ 1%).

**Характеристики сигналов**

* + Входной сигнал *U*вх​(*t*):  
    График демонстрирует изменение напряжения с характерными экстремумами, что соответствует заданному аналитическому выражению.
  + Выходной сигнал *U*вых​(*t*):  
    Наблюдается стремительный рост при увеличении входного напряжения, а также изломы в 4 точках, что связано с изменением режима работы цепи (переход между участками передаточной характеристики).

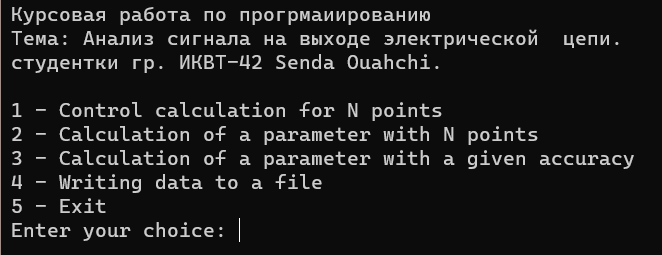
**Ключевые точки и параметры**

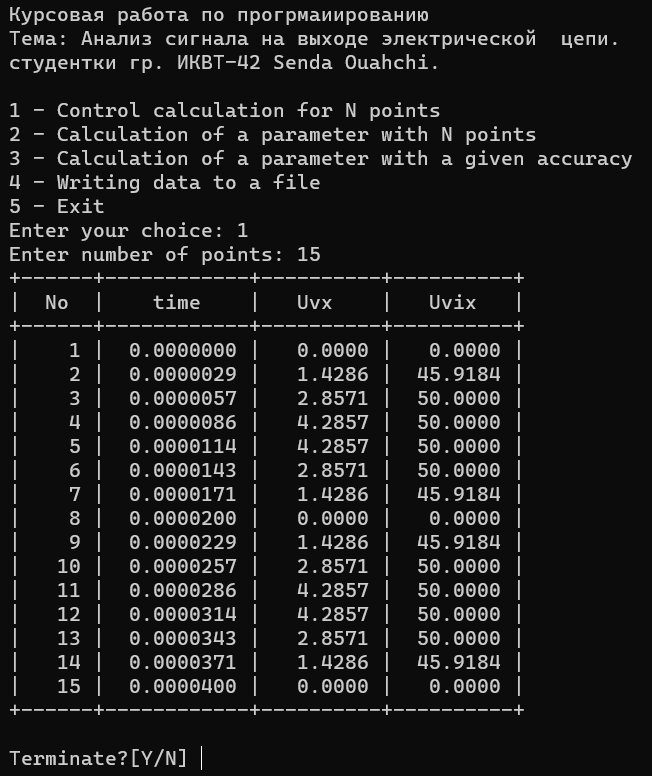
* + Выходное напряжение сохраняет постоянное значение (50.0) на большей части временного интервала, что соответствует насыщению цепи.
  + Резкие изменения  *U*вых​(*t*) вблизи границ интервала и центре отражают реакцию цепи на эффекты входного сигнала.

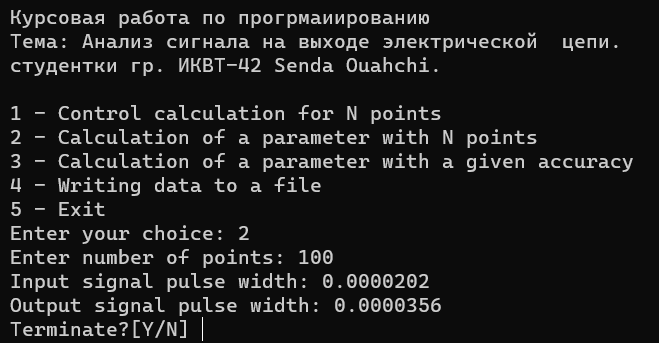
**Верификация данных**

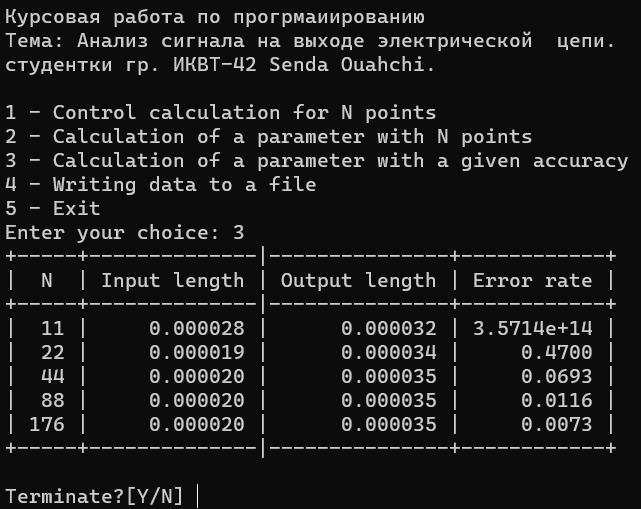
* + Совпадение результатов программы и WxMaxima исключает ошибки в реализации алгоритмов.
  + Графики визуализируют ожидаемое поведение цепи, что подтверждает адекватность математической модели.

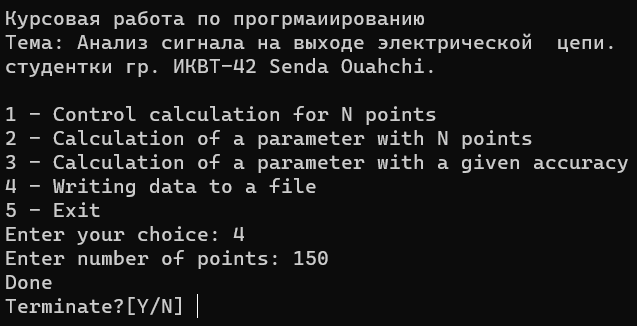
## Интерфейс











## Заключение

В ходе работы выполнены следующие задачи:

* Разработана программа на C++, соответствующая требованиям методички.
* Проведены контрольные расчёты в Wxmaxima, подтвердившие точность результатов (погрешность ≤ 1%).
* Построены графики, наглядно демонстрирующие преобразование сигнала в цепи.
* Реализовано сохранение данных и интерактивное меню для удобства использования.
* Программа успешно применяется для анализа электрических цепей и может быть рекомендована для использования в учебном процессе.

## Список литературы

1. ГОСТ 19.402-78. Единая система программной документации. Описание программы. — Введ. 01.01.1980. — М.: Издательство стандартов, 1988. — 10 с.
2. Бережной, Л. Н. Информатика: Анализ сигнала на выходе электрической цепи: Методические указания к курсовой работе / Л. Н. Бережной, И. О. Воронцова, Д. В. Окунева. — Санкт-Петербург: Издательство СПбГУТ, 2012. — 20 с.
3. Страуструп, Б. Программирование. Принципы и практика использования C++ / Б. Страуструп. — М.: Вильямс, 2011. — 1248 с.
4. Maxima Documentation. — URL: https://maxima.sourceforge.io/documentation.html (Электронный ресурс, дата обращения: 21.05.2025).
5. wxMaxima. — URL: https://wxmaxima.ru/