МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Санкт-Петербургский государственный университет телекоммуникаций

им. проф. М.А. Бонч-Бруевича»

Факультет Инфокоммуникационных сетей и систем

Кафедра Программной инженерии и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине:

**«Программирование»**

тема: Анализ сигнала на выходе электрической цепи

Передаточная характеристика – \_\_\_ вариант

Входной сигнал – \_\_\_ вариант

Выполнил студент(ка):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О., № группы) (подпись)*

Дата выполнения:

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г

Проверил(а):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О.) (подпись)*

Санкт-Петербург

2024

АННОТАЦИЯ

Программный продукт «Консольный анализатор сигналов электрических импульсов.» был разработан и протестирован в результате выполнения курсовой работы «Анализ сигнала на выходе электрической цепи».

Программа представляет собой набор скриптов и исходных кодов, написанных на языке программирования C, предназначенных для создания, анализа и визуализации данных, сгенерированных с использованием определенного алгоритма.

Основное функциональное назначение программы - вычисление параметров, основанных на входных данных, и предоставление пользователю информации о результатах вычислений. Для этого программа выполняет несколько шагов: сначала создает и заполняет файлы с данными, затем анализирует эти данные, вычисляет необходимые параметры и выводит результаты на экран.

Программа также обладает возможностью автоматического создания массивов данных, запуска других программ (wxMaxima) для создания графиков, а также обеспечивает пользовательский интерфейс в виде текстового меню для управления функционалом программы.

СОДЕРЖАНИЕ

[1. Общие сведения 5](#_Toc168131102)

[1.1. Обозначение и наименование программы 5](#_Toc168131103)

[1.2. Программное обеспечение, необходимое для функционирования программы 5](#_Toc168131104)

[1.3. Языки программирования, на которых написана программа 5](#_Toc168131105)

[2. Функциональное назначение 6](#_Toc168131106)

[3. Описание логической структуры 7](#_Toc168131107)

[3.1. Алгоритм программы 7](#_Toc168131108)

[3.1.1 Функция «print\_splash\_screen» 7](#_Toc168131109)

[3.1.2. Функция «main.c» 8](#_Toc168131110)

[3.1.3. «Выбор 1 – введение количества точек» 9](#_Toc168131111)

[3.1.4. «Выбор 2 – расчёт параметра» 9](#_Toc168131112)

[3.1.5. «Выбор 3 – расчёт параметра с заданной точностью» 10](#_Toc168131113)

[3.1.6. «Выбор 4 – ввод чисел в файл» 10](#_Toc168131114)

[3.1.7. «Выбор 5 – вывод исходных данных» 11](#_Toc168131115)

[3.1.8. «Выбор 5 – выход из программы» 11](#_Toc168131116)

[3.2. Используемые методы 12](#_Toc168131117)

[3.3. Структура программы с описанием функций составных частей и связи между ними 12](#_Toc168131118)

[Связи программы с другими программами 13](#_Toc168131119)

[4. Используемые технические средства 14](#_Toc168131120)

[5. Вызов и загрузка 15](#_Toc168131121)

[5.1. Способ вызова программы с соответствующего носителя данных 15](#_Toc168131122)

[5.2. Входные точки в программу 15](#_Toc168131123)

[6. Входные данные 16](#_Toc168131124)

[6.1. Характер, организация и предварительная подготовка входных данных 16](#_Toc168131125)

[6.2. Формат, описание и способ кодирования входных данных 16](#_Toc168131126)

[7. Выходные данные 17](#_Toc168131127)

[7.1. Характер и организация выходных данных 17](#_Toc168131128)

[7.2. Формат, описание и способ кодирования выходных данных 17](#_Toc168131129)

[8. Перечень принятых сокращений 18](#_Toc168131130)

[Выводы 19](#_Toc168131131)

1. Общие сведения
   1. Обозначение и наименование программы

Наименование программы – «Консольный анализатор сигналов электрических импульсов».

Обозначение программы – «КАСЭИ».

* 1. Программное обеспечение, необходимое для функционирования программы

Операционная система: Linux (например Debian).

Консоль или терминал: для компиляции и запуска программы необходим доступ к консоли или терминалу операционной системы.

Пакет wxMaxima: для создания графиков по точкам программа использует пакет wxMaxima.

Для запуска скрипта start.sh также требуется оболочка командной строки Bash.

* 1. Языки программирования, на которых написана программа

Код программы написан на Си и Bash.

1. Функциональное назначение

Функциональное назначение программы состоит в создании, обработке и анализе данных, а также визуализации результатов на графиках. Программа выполняет следующие функции:

1. Позволяет пользователю выбрать количество точек, которые будут записаны в файлы (massiv\_t.txt, massiv\_Uvx.txt, massiv\_Uvix.txt)
2. Запись данных в файл: функция «write\_data\_to\_files()» записывает массивы t[], Uvx[] и Uvix[] в файлы «massiv\_Uvx.txt», «massiv\_t.txt», «massiv\_Uvix.txt».
3. Обработка данных: функция «find\_time\_80V» используя массивы t[],Uvx[], Uvix[], вычисляет определенный параметр на основе заданных алгоритмов и выводит результат в консоль. Текущий параметр показывает значение время, когда функция достигает значения 80V.
4. Вычисление с заданной погрешностью: функция «calculate\_parameter\_with\_precision()» проводит анализ данных с заданной погрешностью, округляя исходный параметр до заданного знака после запятой
5. Визуализация результатов: с использованием скрипта «menu.sh» пользователю предоставляется возможность выбора различных действий, включая построение графиков по точкам на основе данных из файлов.
6. Построение графиков: программа в конце своего выполнения влечёт за собой открытие wxMaxima (курируется файлов «menu.sh», где можно начертить графики по указанным точкам.

Область применения: автоматизированный расчет параметров в научных и инженерных задачах.

1. Описание логической структуры

В данном блок будет описана логическая структура каждой из функции, а также порядок выполнения программы для корректной работы. Здесь представлены блок схемы по ключевым функциям, участвующим в выполнении.

* 1. Алгоритм программы

Эта функция отвечает за начальный вывод заставки, содержащей информацию о курсовой работе и выполняющем её.

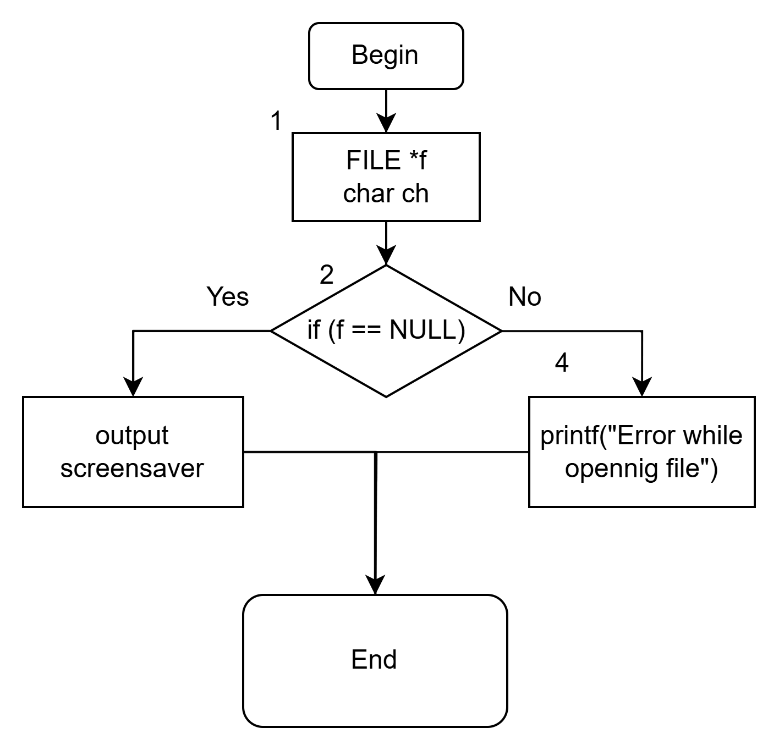
* + 1. Функция «print\_splash\_screen»

Рисунок 1 – Блок-схема функции «print\_splash\_screen»

3.1.2. Функция «main.c»

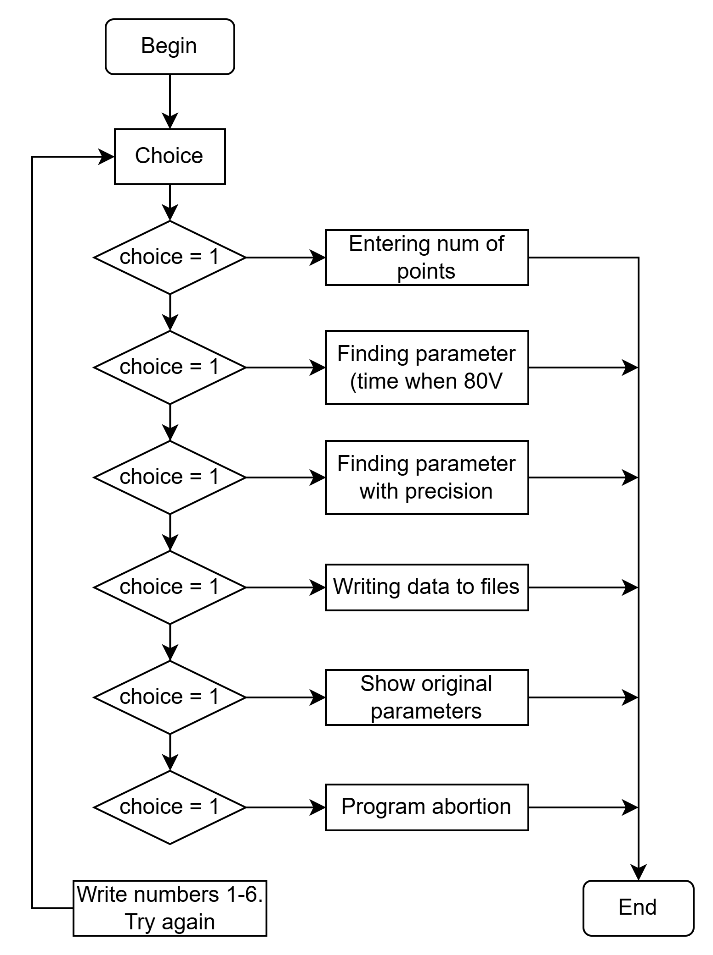
Функция main является главной в программе курсовой работы, ведь отвечает за функционирование меню, а также за выход из программы и запуск всех функций.

Рисунок 2 – Блок-схема функции «main.c»

3.1.3. «Выбор 1 – введение количества точек»

Необходимо ввести количество точек, а также должна выводится ошибка в случае, когда вводится неправильно значение (например, когда пользователь вводит символы или отрицательные числа).

3.1.4. «Выбор 2 – расчёт параметра»

Здесь представлена функция, отвечающая за поиск параметра времени, когда функция достигает значения в 80 Вольт, что является важным для будущего подсчёта этого же параметра с повышенной точносью.

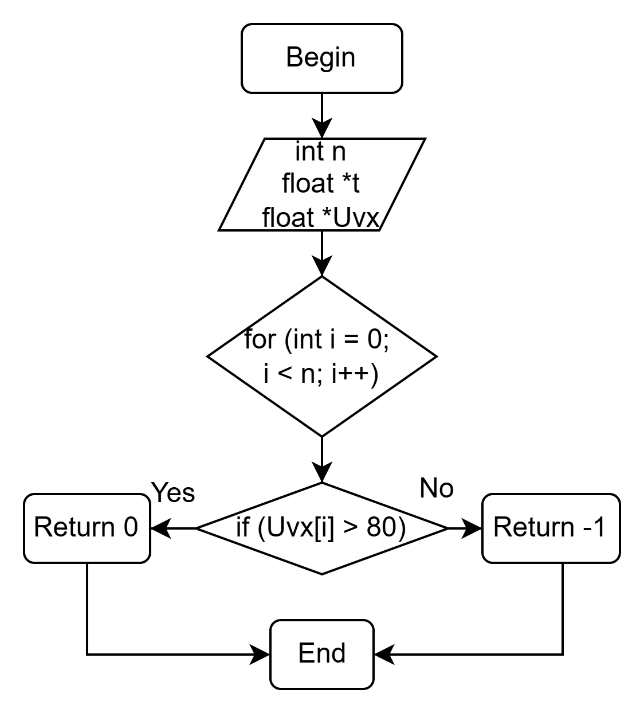


Рисунок 3 – Блок-схема функции find\_time\_80V

3.1.5. «Выбор 3 – расчёт параметра с заданной точностью»

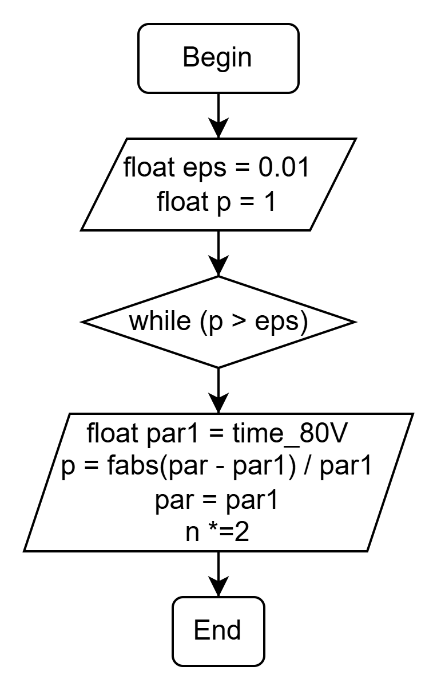
****Здесь пользователь может рассчитать параметр из 3-го пункта меню, однако с повышенной точностью. В данном случае мы будем округлять значение функции, достигаемое ей при значении 80 Вольт.

Рисунок 4 – Блок-схема функции **– «**calculate\_parameter\_with\_precision»

3.1.6. «Выбор 4 – ввод чисел в файл»

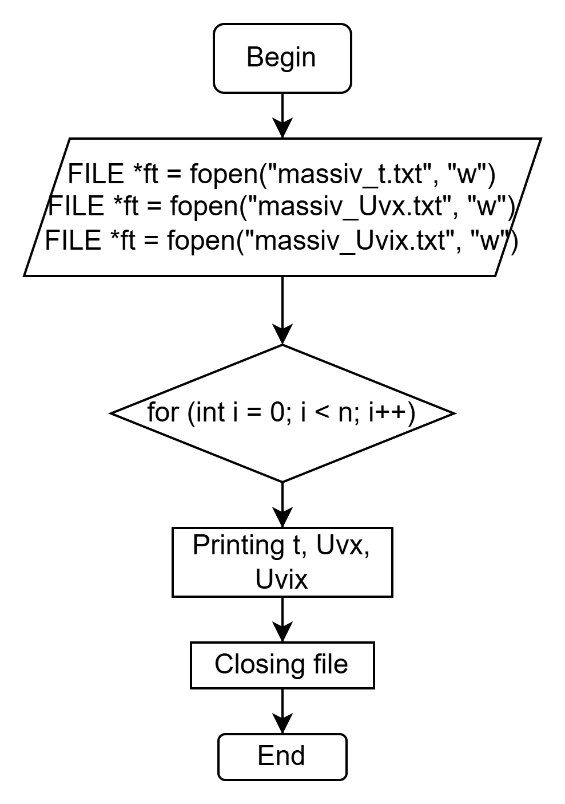
****В данной функции числа, полученные из 1-й опции меню, будут переписываться в три файла: “massiv\_t.txt”, “massiv\_Uvx.txt”, “massiv\_Uvix.txt”, после чего по этим числам можно будет построить графики.

Рисунок 4 – Блок-схема функции «write\_data\_to\_file»

3.1.7. «Выбор 5 – вывод исходных данных»

Здесь формируется таблица, где выводят исходные значения перемнных a, b, t1, tn и tk на случай, если пользователь хочет увидеть их в табличном виде.

3.1.8. «Выбор 6 – выход из программы»

У пользователя есть возможность выйти из программы путём выбора этой опции. При этом запуститься wxMaxima, где будут прорисованны оба графика с предварительно занесёнными из трёх файлов числами.

* 1. Используемые методы

Методы для работы с динамической памятью (malloc, free).

Методы работы с файлами (fopen, fclose, fprintf, fscanf).

Методы математических вычислений (fabs).

* 1. Структура программы с описанием функций составных частей и связи между ними

Программа (main.c) содержит главную функцию «main», которая управляет основным потоком выполнения.

Функция «print\_splash\_screen» выводит первоначальную заставку на экран, где написанные данные об исполнителе курсовой работы.

Функции «form\_t», «form\_Uvx», «form\_Uvix» инициализируют соответствующие массивы для будущей работы.

Функция «form\_tabl» отвечает за визуальный вывод данных таблицы со значениями t, Uvx, Uvix, а также порядкого номера измерения.

Функция «write\_data\_to\_files» в программе «calculations.c» используются для чтения данных из файлов и загрузки их в массивы.

Функция «find\_time\_80V» находит время, при котором Uvx достигает значения в 80V.

Функция «control\_calculation» контролирует порядок выполнения.

Функция «calculate\_parameter\_with\_precision» высчитывает параметр с заданной точностью.

Программа «show\_parameters» выводит пользователю значения исходных параметров (a, b, t1, tn, tk).

Связи программы с другими программами

Программа «main.c» и скрипт «menu.sh» работают в тесной связи для обеспечения удобного использования и управления функциональностью анализатора сигналов. Вот как они взаимодействуют:

«menu.sh» выполняет роль главного меню программы. При запуске программы пользователю предоставляется выбор из 6 опций, представленных в главном меню. Вот что делает каждая из них:

* + - 1. Ввод количества точек;
      2. Поиск параметра (время при значении функции 80В);
      3. Поиск параметра с заданной точностью;
      4. Запись чисел в файлы;
      5. Вывод исходных параметров;
      6. Выход.

Важное замечание состоит в том, что порядок выполнения программы условно ограничен, ведь пользователь в самом начале может выбрать только опции 1, 5 и 6. В иных случаях выведется ошибка.

После выполнения опции 1 у пользователя появляется возможность выбрать опцию 2, чтобы посчитать параметр, а также опцию 4, чтобы записать числа в файл. После этого уже можно посчитать параметр с заданной точностью под опцией 3.

Подразумевается, что пользователь будет выполнять программу в следующем порядке: 5, 1, 2, 3, 4, 6.

После выполнения программы открывается wxMaxima, где уже считаны значения из 3-х файлов, пользователь же может построить графики по ним.

1. Используемые технические средства

Языки программирования: C, Bash.

Среда выполнения: ОС Linux.

Среда разработки / компилятор: GCC.

Версия используемого компилятора: GCC 9.3.0.

Средства ввода и вывода: стандартные потоки ввода-вывода (stdin, stdout), файлы.

1. Вызов и загрузка
   1. Способ вызова программы с соответствующего носителя данных

Программа может быть вызвана из командной строки операционной системы Linux с помощью команды «./menu.sh».

После запуска команды пользователь увидит интерактивное меню с возможными действиями. Он может выбрать один из предложенных пунктов, включая введение количества точек или вывод исходных параметров.

* 1. Входные точки в программу

Пользователь может взаимодействовать с программой, выбирая различные опции из меню скрипта «menu.sh», который компилирует все необходимые файлы в один исполняемый «main.exe».

Входные точки также включают файлы данных, такие как «massiv\_t.txt», «massiv\_Uvx.txt» и «massiv\_Uvix.txt», которые содержат значения времени и соответствующие значения параметров Uvx и Uvix. Эти файлы используются программой для чтения данных и выполнения расчетов параметров.

1. Входные данные
   1. Характер, организация и предварительная подготовка входных данных

Программа принимает входные данные в виде временных точек и соответствующих значения параметров Uvx и Uvix.

Эти данные должны быть предварительно подготовлены и записаны в текстовые файлы «massiv\_t.txt», «massiv\_Uvx.txt» и «massiv\_Uvix.txt» соответственно.

Входные данные должны содержать числовые значения, где time.txt содержит временные точки, Uvx.txt содержит значения параметра Uvx, а Uvix.txt содержит значения параметра Uvix.

* 1. Формат, описание и способ кодирования входных данных

Файл «massiv\_t.txt» содержит временные точки, каждая из которых записана на отдельной строке в виде числа с плавающей запятой.

Файлы «massiv\_Uvx.txt» и «massiv\_Uvix.txt» содержат значения параметров Uvx и Uvix соответственно, каждое значение записано на отдельной строке в виде числа с плавающей запятой.

Для удобства чтения данных программой и обеспечения совместимости с форматом файла, данные должны быть записаны в текстовом формате без дополнительного форматирования или кодирования.

Программа использует функцию fprintf для чтения числовых значений из файлов, что предполагает использование стандартного текстового формата для входных данных.

1. Выходные данные
   1. Характер вывода данных и их организация

Программа выводит результаты расчетов в консоль.

Каждый результат представляет собой значение выбранного параметра или приведенный результат расчёта параметра с заданной точностью.

Для каждого набора точек результаты выводятся в отдельно.

* 1. Формат, описание и способ кодирования выходных данных

Результаты выводятся в текстовом формате.

Для расчёта параметра с заданной точностью каждая строка содержит кол-во точек, значение параметра и округлённое значение параметра.

Значение параметра и погрешность выводятся в формате числа с плавающей запятой.

1. Перечень принятых сокращений

BASH – Bourne Again SHell (Обновленная оболочка Борна)

GCC – (GNU Compiler Collection)

Uvx – входное напряжение цепи

Uvix – выходное напряжение цепи

Выводы

В итоге курсовой работы по созданию программного продукта "Консольного анализатора сигналов электрических импульсов" было достигнуто множество важных результатов:

Разработан полный набор скриптов и исходных кодов на языках программирования Си и Bash для создания, анализа и визуализации данных сигналов.

Основным функционалом программы стало вычисление параметров на основе входных данных и предоставление информации о результатах пользователю.

Программа имеет возможность автоматического создания массивов данных, запуска графических программ для визуализации и текстовое меню для управления ее функциями.

Логическая структура программы четко описана с примерами алгоритмов и использованных методов, что облегчает восприятие и модификацию кода.

Программа показала эффективность в обработке различных данных, а также автоматизации расчетов параметров.

Технические средства, использованные в работе, включают компилятор GCC, языки программирования Си и Bash, операционную систему Linux и пакет wxMaxima для построения графиков.

Входные данные корректно организованы в файлах текстового формата, содержащих точки времени и соответствующие значения параметров, что обеспечивает понятность и удобство обработки программой.

#### **ПРИЛОЖЕНИЕ А (BASH-файл «menu.sh»)**

**#!/bin/bash**

**# Компиляция программы на языке C**

**gcc main.c -o main.exe**

**# Запуск программы на языке C**

**./main.exe**

**# Запуск программы wxmaxima**

**wxmaxima graphics.wxmx**

**# Завершение скрипта**

**exit 0**

#### **ПРИЛОЖЕНИЕ Б (c-файл «main.c»)**

**// main.c**

**#include "calculations.h"**

**#include "calculations.c"**

**int main() {**

**int calculation\_done = 0;**

**int time\_80V\_calculated = 0;**

**int parameter\_calculated = 0;**

**float t[N], Uvx[N], Uvix[N];**

**int n = 0;**

**float time\_80V = 0; // Variable to store time when Uvx reaches 80V**

**print\_splash\_screen(); // Display splash screen**

**while (1) {**

**int choice;**

**// Print top border**

**printf("\n\n\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\*\033[34m||||||||||||||||||\033[0mMenu\033[34m|||||||||||||||||||\033[0m\*\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\* 1 -\033[32m Control calculation for 'n' points \033[0m\*\n");**

**printf("\* 2 -\033[32m Find time when Uvx reaches 80V \033[0m\*\n");**

**printf("\* 3 -\033[32m Calculate parameter with precision \033[0m\*\n");**

**printf("\* 4 -\033[32m Write data to files \033[0m\*\n");**

**printf("\* 5 -\033[32m Show initial parameters \033[0m\*\n");**

**printf("\* 6 -\033[33m Abort program execution \033[0m\*\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf(" Enter your choice: ");**

**// Prompt user for input**

**if (scanf("%d", &choice) != 1 || choice < 1 || choice > 6) {**

**printf("\033[1;31mInvalid choice. Please enter a valid option (1, 2, 3, 4, 5 or 6): \033[0m");**

**while (getchar() != '\n'); // Clear input buffer**

**continue;**

**}**

**printf("\n");**

**switch(choice) {**

**case 1:**

**printf("\n\n Enter the number of points: ");**

**while (1) {**

**if (scanf("%d", &n) == 1 && n > 0) {**

**control\_calculation(n, t, Uvx, Uvix, &calculation\_done);**

**parameter\_calculated = 0; // Reset parameter calculation flag**

**time\_80V\_calculated = 0; // Reset time 80V calculation flag**

**break; // Break if valid input**

**} else {**

**printf("\033[1;31mInvalid input. Please enter a positive integer for the number of points:\033[0m ");**

**while (getchar() != '\n'); // Clear input buffer**

**}**

**}**

**break;**

**case 2:**

**if (calculation\_done) {**

**time\_80V = find\_time\_80V(n, t, Uvx);**

**if (time\_80V != -1) {**

**printf(" Time when 'Uvx' reaches 80V: %f\n", time\_80V);**

**time\_80V\_calculated = 1; // Mark time 80V calculation as done**

**} else {**

**printf("Uvx never reaches 80V.\n");**

**}**

**} else {**

**printf("\033[1;31mNo data available. Please perform a control calculation first.\033[0m\n");**

**}**

**break;**

**case 3:**

**if (time\_80V\_calculated) {**

**calculate\_parameter\_with\_precision(n, t, Uvx, Uvix, time\_80V); // Pass time when Uvx reaches 80V**

**parameter\_calculated = 1; // Set parameter calculation flag**

**} else {**

**printf("\033[1;31mTime when Uvx reaches 80V has not been calculated. Please perform this calculation first.\033[0m\n");**

**}**

**break;**

**case 4:**

**if (calculation\_done) {**

**write\_data\_to\_files(n, t, Uvx, Uvix);**

**} else {**

**printf("\033[1;31mNo data to write. Perform control calculation first.\033[0m\n");**

**}**

**break;**

**case 5:**

**show\_parameters(); // Display calculation parameters**

**break;**

**case 6:**

**printf("\033[1;33mProgram execution aborted by user.\033[0m\n");**

**return 0; // Exit the program**

**}**

**// Clear input buffer**

**while (getchar() != '\n');**

**}**

**return 0;**

**}**

#### **ПРИЛОЖЕНИЕ В (c-файл «calculations.c»)**

**// calculations.c**

**#include "calculations.h"**

**// Функция для вывода заставки**

**void print\_splash\_screen() {**

**FILE \*f = fopen("screensaver.txt", "r");**

**if (f == NULL) {**

**printf("Error opening file.\n");**

**return;**

**}**

**char ch;**

**while ((ch = fgetc(f)) != EOF) {**

**putchar(ch);**

**}**

**fclose(f);**

**}**

**// Описание функции формирования массива времени form\_t**

**void form\_t(int n, float \*t, float \*dt) {**

**float tn = 10, tk = 35;**

**\*dt = (tk - tn) / (n - 1);**

**for (int i = 0; i < n; i++) {**

**t[i] = tn + i \* (\*dt);**

**}**

**}**

**// Описание функции формирования массива Uvx**

**void form\_Uvx(int n, float \*t, float \*Uvx) {**

**float t1 = 30, a = 12, b = 48;**

**for (int i = 0; i < n; i++) {**

**if (t[i] < t1)**

**Uvx[i] = a \* (t[i] - 10);**

**else**

**Uvx[i] = a \* (t1 - 10) - b \* (t[i] - t1);**

**}**

**}**

**// Описание функции формирования массива Uvix**

**void form\_Uvix(int n, float \*Uvx, float \*Uvix) {**

**float Uvx1 = 5, Uvx2 = 25, U1 = 20, U2 = 150;**

**for (int i = 0; i < n; i++) {**

**if (Uvx[i] < Uvx1)**

**Uvix[i] = U1;**

**else if (Uvx[i] <= Uvx2)**

**Uvix[i] = 6.5 \* Uvx[i] - 12.5;**

**else**

**Uvix[i] = U2;**

**}**

**}**

**// Описание функции вывода данных в виде таблицы form\_tabl**

**void form\_tabl(int n, float \*t, float \*Uvx, float \*Uvix) {**

**printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\* \033[32mNum\033[0m | \033[32m t \033[0m| \033[32mUvx \033[0m | \033[32m Uvix \033[0m \*\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**for (int i = 0; i < n; i++) {**

**printf("\* %3d | %6.2f | %8.2f | %8.2f \*\n", i + 1, t[i], Uvx[i], Uvix[i]);**

**}**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**}**

**// Описание функции записи данных в файлы**

**void write\_data\_to\_files(int n, float \*t, float \*Uvx, float \*Uvix) {**

**FILE \*ft = fopen("massiv\_t.txt", "w");**

**FILE \*fUvx = fopen("massiv\_Uvx.txt", "w");**

**FILE \*fUvix = fopen("massiv\_Uvix.txt", "w");**

**for (int i = 0; i < n; i++) {**

**fprintf(ft, "%f\n", t[i]);**

**fprintf(fUvx, "%f\n", Uvx[i]);**

**fprintf(fUvix, "%f\n", Uvix[i]);**

**}**

**fclose(ft);**

**fclose(fUvx);**

**fclose(fUvix);**

**printf(" Data written to files.\n");**

**}**

**// Функция для нахождения времени, при котором Uvx достигает 80 В**

**float find\_time\_80V(int n, float \*t, float \*Uvx) {**

**for (int i = 0; i < n; i++) {**

**if (Uvx[i] > 80) {**

**return t[i];**

**}**

**}**

**return -1; // Возвращаем -1, если Uvx не достигает 80 В**

**}**

**// Функция для контрольного расчета для n точек**

**void control\_calculation(int n, float \*t, float \*Uvx, float \*Uvix, int \*calculation\_done) {**

**float dt;**

**form\_t(n, t, &dt);**

**form\_Uvx(n, t, Uvx);**

**form\_Uvix(n, Uvx, Uvix);**

**form\_tabl(n, t, Uvx, Uvix);**

**\*calculation\_done = 1;**

**}**

**void calculate\_parameter\_with\_precision(int n, float \*t, float \*Uvx, float \*Uvix, float time\_80V) {**

**float eps = 0.01; // Заданная погрешность**

**float p = 1; // Текущая погрешность**

**while (p > eps) {**

**float par1 = time\_80V; // Используем time\_80V в качестве параметра**

**p = fabs(par - par1) / par1;**

**printf("\033[32mn:\033[0m %d, \033[32mparameter:\033[0m %f, \033[32mprecision (until 0.01):\033[0m %.2f.\n", n, par1, par1); // Округляем параметр до двух знаков после запятой**

**par = par1;**

**n \*= 2;**

**}**

**}**

**// Функция для вывода параметров расчета**

**void show\_parameters() {**

**float a = 12, b = 48, t1 = 30, tn = 10, tk = 35;**

**printf(" \033[34mInitial parameters:\033[0m ");**

**printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf(" \* \033[32mParameter\033[0m | \033[32m Value \033[0m \*\n");**

**printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf(" \* a | %8.2f \*\n", a);**

**printf(" \* b | %8.2f \*\n", b);**

**printf(" \* t1 | %8.2f \*\n", t1);**

**printf(" \* tn | %8.2f \*\n", tn);**

**printf(" \* tk | %8.2f \*\n", tk);**

**printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**}**

#### **Приложение Г (header-файл «calculations.h»)**

**// calculations.h**

**#ifndef CALCULATIONS\_H**

**#define CALCULATIONS\_H**

**#include <stdio.h>**

**#include <math.h>**

**#include <ctype.h>**

**#define N 1000**

**void print\_splash\_screen();**

**void form\_t(int n, float \*t, float \*dt);**

**void form\_Uvx(int n, float \*t, float \*Uvx);**

**void form\_Uvix(int n, float \*Uvx, float \*Uvix);**

**void form\_tabl(int n, float \*t, float \*Uvx, float \*Uvix);**

**void write\_data\_to\_files(int n, float \*t, float \*Uvx, float \*Uvix);**

**float parameter(int n, float \*t, float \*Uvx, float \*Uvix);**

**void control\_calculation(int n, float \*t, float \*Uvx, float \*Uvix, int \*calculation\_done);**

**void calculate\_parameter\_with\_precision(int n, float \*t, float \*Uvx, float \*Uvix, float time\_80V);**

**float find\_time\_80V(int n, float \*t, float \*Uvx);**

**void show\_parameters();**

**#endif // CALCULATIONS\_H**