SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ELECTRICITY BILL CALCULATOR

CSE23CL101 – Programming in C Laboratory

Submitted by

NAME AND UNIQUE ID:

ABEENAYA P

E0325109

In partial fullfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Sri Ramachandra Faculty of Engineering and Technology, Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

November, 2025

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified to be the bonafide record of the work done by ----------

---------- of Semester I, First Year B.Tech Degree course in Sri Ramachandra Faculty of Engineering and Technology, of Computer Science and Engineering Department in the **CSE23CL101 Programming in C Laboratory** during the academic year **2025-2026.**

SIGNATURE OF EXAMINER:

MS. ANUSHIYA J,

Assistant Professor,

Department of Artificial Intelligence and Data Analytics,

Sri Ramachandra Faculty of Engineering and Technology,

Sri Ramachandra Institute of Higher Education and Research,

Porur, Chennai, Tamil Nadu.

Submitted for the Programming in C Laboratory project presentation held at Sri Ramachandra Faculty of Engineering and Technology on ----------

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLES OF CONTENT

ABSTRACT

The Electricity Bill Calculator is a software application developed to automate the process of calculating monthly electricity bills for consumers. The syst1em

accepts user input such as customer details and units of electricity consumed, then applies predefined tariff rates using conditional logic to compute the total payable amount. It includes features for detailed charge breakdown, validation of inputs to ensure accuracy, and file handling capabilities to store and retrieve billing records. Additionally, the project incorporates functionality to track previous bills and outstanding payments, improving record management. A simple text-based graphical display provides users with a clear visual representation of their electricity consumption and charges. This project aims to reduce manual errors, improve efficiency in bill generation, and offer an easy-to-use interface for both consumers and administrators.

CHAPTER 1

INTRODUCTION

1.1 Overview

The Electricity Bill Calculator is a C programming–based application developed to simplify and automate the process of generating electricity bills for consumers. The project focuses on capturing user data such as customer name, ID, and units consumed, and then calculating the total bill amount based on predefined tariff rates. It also includes features to handle outstanding payments and maintain billing history through file handling.

This system ensures accuracy and efficiency by validating user inputs and applying logical conditions for tariff calculation. It provides a detailed breakdown of charges, helping users understand how their total bill is computed. Additionally, the project incorporates a simple text-based graphical display to visually represent consumption trends.

By integrating important programming concepts such as structures, conditional statements, loops, file handling, and data validation, this project not only automates billing operations but also serves as an excellent example of applying core C programming techniques to real-world problems.

1.2 Motivation

Electricity billing is a routine yet critical process that affects every household and business. Manual calculation of bills can lead to errors, delays, and confusion, especially when dealing with varying tariff rates and large volumes of data.

This project was motivated by the desire to create a practical solution that simplifies electricity bill computation using programming concepts. By developing an Electricity Bill Calculator, we aimed to apply our knowledge of conditional logic, file handling, and user input validation in a real-world context. The project also allowed us to explore how software can improve accuracy, transparency, and user experience in utility services.

Furthermore, this system serves as a foundational step toward building more advanced billing applications, and it helped us strengthen our technical skills while solving a meaningful problem.

1.3 Problem Statement

This project aims to solve these issues by developing a simple and reliable Electricity Bill Calculator that automates the billing process based on units consumed.

- ✓ DEFINE THE STRUCTURE FOR BILL DETAILS
- ✓ ACCEPT USER INPUT FOR UNITS CONSUMED

- ✓ APPLY CONDITIONAL LOGIC FOR TARIFF CALCULATION
- ✓ DISPLAY BREAKDOWN OF CHARGES
- ✓ INTEGRATE FILE HANDLING FOR BILL RECORDS
- ✓ ADD FEATURES FOR TRACKING PREVIOUS BILLS AND OUTSTANDING PAYMENTS
- ✓ VALIDATE USER INPUT FOR DATA ACCURACY
- ✓ INCLUDE A GRAPHICAL DISPLAY OPTION USING SIMPLE TEXT-BASED CHARTS

By addressing these requirements , The project highlights the practical problem-solving skills, attention to detail, and the ability to design robust and error-free programs**.**

1.4 Objectives

The main objectives of this project are:

- To automate the calculation of electricity bills based on the number of units consumed using conditional tariff logic.
- To reduce manual errors in billing by implementing a reliable and consistent software solution.
- To provide a detailed breakdown of charges, including base cost, taxes, and total payable amount.

- To implement file handling for storing and retrieving customer billing records.

- To enable tracking of previous bills and outstanding payments for better record management.

- To validate user input to ensure data accuracy and prevent invalid entries.

- To enhance user experience by displaying results in a clear format, including simple text-based graphical charts.

- To apply programming concepts such as control structures, data structures, and file operations in a real-world application.

CHAPTER 2

Core Programming Concepts Used:

This Electricity Bill Calculator program demonstrates several core programming concepts in C.

2.1 DATA STRUCTURES:

Use of a struct (Bill) to store aggregated customer bill details (name, consumer ID, units, amounts). This custom structure to group related data promotes organized and modular data handling.

2.2 INPUT VALIDATION:

Functions (getValidatedInt, getValidatedFloat) repeatedly read user input until valid positive integers or non-negative floats are entered.

2.3 CONDITIONAL LOGIC:

The tariff calculation employs if-else if-else statements to apply different rates based on units consumed, illustrating control flow.

2.4 LOOPS:

loops for input validation and reading the file contents for outstanding balance, and for loops for drawing the usage graph.

2.5 FUNCTIONS:

Modular design with separate functions for input, calculating bills, displaying information, managing files, and graph display promotes code reuse and clarity.

2.6 FILE HANDLING:

fopen, fprintf, fscanf, and fclose used to save bill records to a file (bills.txt) and to retrieve previous outstanding dues

2.7ASCII- BASED GRAPHICAL DISPLAY:

Visual representation of electricity usage using * characters and to enhances user experience with a simple chart.

2.8ARITHMETIC OPERATIONS:

Calculations of tariff amounts and totals using multiplication and addition.

2.9FORMATTED OUTPUT:

printf() used with formatting specifiers (%.2f, %d) for clean display.

2.10CODE MODULARITY AND READABILITY:

Logical separation of tasks into functions and Use of comments and meaningful names improves maintainability.

CHAPTER 3

METHODOLOGY

The project was developed using a Structured Programming approach in the C language, following a clear, step-by-step process to ensure all requirements were met, including calculation accuracy, input validation, and data persistence.

3.1 PROJECT ANALYSIS:

Understand the problem and considered real-world requirements such as tiered tariff rates, outstanding dues, and persistent storage.

3.2 DESIGNING AND PLANNING:

Plan the program structure to encapsulate all relevant customer and billing data. Outlined the program flow: input → calculation → display → storage.

3.3 IMPLEMENTATION:

Developed the program in C using standard libraries (stdio.h, stdlib.h, string.h).

- Implemented:
- Input validation functions to ensure correct data entry.
- Tariff calculation logic using conditional statements.
- File handling to store and retrieve billing records.
- ASCII-based graph to visually represent electricity usage.

3.4 TESTING AND VALIDATION:

Tested the program with various input scenarios:

- Valid and invalid consumer IDs
- Different unit ranges (e.g., below 100, between 100–300, above 300)
- Repeated entries to verify outstanding dues tracking and Ensured correct file writing and reading operations.

3.5 USER INTERFACE AND OUTPUT:

Usage graph using stars (*) for visual feedback and Ensured messages guide the user through each step.

3.6 DOCUMENTATION:

Added comments and structured code for readability.Prepared flowcharts and presentation slides to explain logic and features.Documented all programming concepts and future enhancement ideas.

CHAPTER 4

IMPLEMENTATION AND RESULTS

CODE:

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// STEP 1: Structure to store bill details
struct Bill {
    char customerName[50];
    int customerID;
    float units;
    float outstanding;
    float energyCharge;
    float fixedCharge;
    float tax;
    float totalAmount;
};

// Function to calculate energy charge based on slabs
float calculateEnergyCharge(float units) {
    float charge = 0;

    if (units <= 100)
        charge = units * 1.50;
    else if (units <= 200)
        charge = (100 * 1.50) + ((units - 100) * 2.00);
    else
        charge = (100 * 1.50) + (100 * 2.00) + ((units - 200) * 3.50);

    return charge;
}
```

```c
// Function to safely get integer input
int getValidatedInt(char *msg) {
    int x;
    while (1) {
        printf("%s", msg);
        if (scanf("%d", &x) == 1) return x;
        else {
            printf("Invalid input! Enter a valid number.\n");
            fflush(stdin);
        }
    }
}
```

main.c Share

```c
// Function to safely get float input
float getValidatedFloat(char *msg) {
    float x;
    while (1) {
        printf("%s", msg);
        if (scanf("%f", &x) == 1) return x;
        else {
            printf("Invalid input! Enter a valid number.\n");
            fflush(stdin);
        }
    }
}
```

```c
// STEP 5: Save bill to file
void saveBillToFile(struct Bill b) {
    FILE *fp = fopen("bill_records.txt", "a");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return;
    }

    fprintf(fp, "Customer: %s | ID: %d | Units: %.2f | Total: %.2f\n",
            b.customerName, b.customerID, b.units, b.totalAmount);

    fclose(fp);
    printf("\nBill saved successfully!\n");
}
```

```c
// STEP 6: Display previous bills
void showPreviousBills() {
    FILE *fp = fopen("bill_records.txt", "r");
    if (fp == NULL) {
        printf("No previous bills found.\n");
        return;
    }

    char line[200];
    printf("\n----- Previous Bill Records -----\n");
    while (fgets(line, sizeof(line), fp)) {
        printf("%s", line);
    }

    fclose(fp);
}

```

```c
// STEP 8: Show units chart
void showUnitsChart(float units) {
    int bars = units / 20; // 1 bar = 20 units

    printf("\n----- Units Consumption Chart -----\n");
    printf("20 units per bar\n");
    printf("[ ");

    for (int i = 0; i < bars; i++)
        printf("#");

    printf(" ]  %.2f units\n", units);
}

```

```c
int main() {

    struct Bill b;

    // STEP 2: Accept User Input
    printf("Enter Customer Name: ");
    scanf("%s", b.customerName);

    b.customerID = getValidatedInt("Enter Customer ID: ");
    b.units = getValidatedFloat("Enter Units Consumed: ");
    b.outstanding = getValidatedFloat("Enter Outstanding Amount: ");

    // STEP 3: Calculate tariff
    b.energyCharge = calculateEnergyCharge(b.units);
    b.fixedCharge = 50;
    b.tax = (b.energyCharge + b.fixedCharge) * 0.05;
    b.totalAmount = b.energyCharge + b.fixedCharge + b.tax + b.outstanding;

    // STEP 4: Display breakdown
    printf("\n----- ELECTRICITY BILL -----\n");
    printf("Customer Name     : %s\n", b.customerName);
```

```c
    printf("Customer ID       : %d\n", b.customerID);
    printf("Units Consumed    : %.2f\n\n", b.units);

    printf("Energy Charges    : %.2f\n", b.energyCharge);
    printf("Fixed Charges     : %.2f\n", b.fixedCharge);
    printf("Tax (5%%)          : %.2f\n", b.tax);
    printf("Outstanding Amt   : %.2f\n", b.outstanding);

    printf("-------------------------------------\n");
    printf("Total Payable     : %.2f\n", b.totalAmount);
    printf("-------------------------------------\n");

    // STEP 7: Units chart
    showUnitsChart(b.units);

    // STEP 5: Save bill to file
    saveBillToFile(b);
```

```
142     // STEP 6: Previous bill tracking
143     int choice = getValidatedInt("\Do you want to see previous bills? (1 = Yes):
            ");
144
145     if (choice == 1) {
146         showPreviousBills();
147     }
148
149     printf("\n--- THANK YOU ---\n");
150
151     return 0;
152 }
```

OUTPUT:

Output

```
Enter Customer Name: elon
Enter Customer ID: 100
Enter Units Consumed: 200
Enter Outstanding Amount: 200

----- ELECTRICITY BILL -----
Customer Name      : elon
Customer ID        : 100
Units Consumed     : 200.00

Energy Charges     : 350.00
Fixed Charges      : 50.00
Tax (5%)           : 20.00
Outstanding Amt    : 200.00
---------------------------------------
Total Payable      : 620.00
---------------------------------------

----- Units Consumption Chart -----
20 units per bar
[ ########## ]  200.00 units
```

OUTPUT EXPLANATION:

The Electricity Bill Calculator generates a detailed and user-friendly output based on the input provided by the user. The output includes the following components:

1. Bill Breakdown

After entering the consumer ID and units consumed, the program displays a structured summary:

- **Consumer ID: Unique identifier for the user.**
- **Units Consumed: Total electricity usage entered.**
- **Outstanding Amount: Previous unpaid dues retrieved from the file.**
- **Current Bill: Charges calculated based on tiered tariff rates.**
- **Total Due: Sum of current bill and outstanding amount.**

This breakdown helps users understand how their total payable amount is derived.

2. Graphical Usage Display

The program includes a simple ASCII-based graph:

- For every 10 units consumed, one * (star) is printed.
- Example: If 120 units are consumed, the graph will show 12 stars → **********

This visual representation gives users a quick sense of their electricity usage.

3. File Storage Confirmation:

Once the bill is calculated:

- The details are saved in a file named bills.txt.
- A confirmation message is displayed: "Bill details saved successfully!"

This ensures that the billing data is stored for future reference and outstanding tracking.

4. Persistent Tracking

If the same consumer ID is used again:

- The program retrieves previous bills from the file.
- It adds the last bill amount to the previous outstanding to compute the new total due.

This feature simulates real-world billing systems where unpaid dues are carried forward.

ALGORITHM:

Step 1: Start

Step 2: Display program title

- Print "=== Electricity Bill Calculator ==="

Step 3: Accept and validate user input

- Prompt for Consumer ID
    - Ensure it is a positive integer
- Prompt for Units Consumed
- Ensure it is a non-negative float

Step 4: Retrieve outstanding dues

- Open bills.txt file in read mode
- Search for matching Consumer ID
- If found, add previous bill amount and outstanding to compute total outstanding

Step 5: Calculate current bill

- Apply tariff logic:
- If units ≤ 100 → ₹1.50/unit
- If 101–300 → ₹1.50 for first 100 + ₹2.00/unit for next 200
- If >300 → ₹1.50 for first 100 + ₹2.00 for next 200 + ₹3.00/unit for remaining

Step 6: Display usage graph

- For every 10 units consumed, print one * (star)

Step 7: Save bill to file

- Open bills.txt in append mode
- Write: ConsumerID UnitsConsumed BillAmount Outstanding

Step 8: Display bill breakdown

- Show Consumer ID, Units Consumed, Outstanding, Current Bill, and Total Due

Step 9: End

- Print confirmation message and terminate program.

Chapter 5

CONCLUSION

The **Electricity Bill Calculator** project successfully achieved its primary objective: To develop a reliable and accurate console application for managing and calculating electricity consumption charges. The implementation effectively utilized core C programming concepts to solve practical problem. This project ensures clarity, maintainability, and ease of future enhancements. The final output provides a clear, detailed breakdown of charges and includes a basic text-based visualization, confirming the program's utility and functionality. Overall, this project not only strengthened our understanding of structured programming but also highlighted the importance of user-friendly interfaces and data accuracy in utility-based applications.

FUTURE WORK

To enhance the functionality, usability, and scalability of the Electricity Bill Calculator, the following improvements can be considered:

1.DYNAMIC TARIFF SYSTEM:

- Implement a flexible tariff structure that can be updated from a configuration file or database.
- Allows adaptation to real-world changes in electricity pricing without modifying the source code.

2. GUI INTEGRATION:

- Develop a graphical user interface using libraries like **GTK+**, **WinBGIm**, or **C++ with Qt**.
- A GUI would improve user interaction and make the application more intuitive for non-technical users.

3. ONLINE BILL PAYMENT STIMULATION:

- Simulate online payment options using mock APIs or file-based transaction logs.
- Include payment confirmation, receipt generation, and basic encryption for sensitive data.

4. ADVANCED DATA VISUALIZATION:

- Use ASCII-based charts or integrate with external tools to display usage trends over time.
- Visualizing monthly or yearly consumption helps users understand and manage their electricity usage.

5. DATABASE INTEGRATION:

- Replace file handling with **SQLite** or **MySQL** to store bill records, user profiles, and payment history.
- This improves data integrity, scalability, and query efficiency.

6.MOBILE AND WEB PORTABILITY:

- Port the application to mobile platforms or convert it into a web-based tool using **CGI with C** or **WebAssembly**.
- This would make the calculator accessible from any device.

7. USER AUTHENTICATION:

- Add login functionality to protect user data and personalize billing records.
- Include role-based access for administrators and consumers.

8. UNIT TESTING AND ERROR LOGGING:

- Implement unit tests for tariff logic, input validation, and file operations.
- Add error logging to track runtime issues and improve debugging.

REFERENCES

- **Programming in ANSI C by E. Balagurusamy**
- **The C Programming Language by Brian W. Kernighan and Dennis M. Ritchie**
- https://youtu.be/0yxWwIac3tc?si=88XLA8NIrcbOUYDB
- https://youtu.be/DfBRMYE7Ofo?si=ZnK-SAPtUjiv2Mib
- LET US C by Yashavant Kanetkar