VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

TECHNICAL TRAINING

Mini project On

" TARIFF CALCULATION "

By

GAGAN DHEEP	**4MT21CS053**
JOYSTON MONTHERO	**4MT21CS061**
AKHIL T	**4MT21CS017**
AKSHAY ACHARYA	**4MT21CS018**
DIYA J A	**4MT21CS051**

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

(*Accredited by NBA*)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution

(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)

Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Badaga Mijar, Moodabidri-574225, Karnataka

2022-23

Project Report: Electricity Tariff Calculator

Table of Contents

1. Abstract

The Electricity Tariff Calculator is a command-line utility designed to compute electricity tariffs for consumers based on their consumer type (domestic or industrial) and the time of day they utilize electricity. This report provides an in-depth overview of the project, including its purpose, scope, key features, and its potential impact on consumers and utility companies.

2. Introduction

2.1 Background

Electricity is a fundamental utility that powers our modern world. Different consumers, such as residential (domestic) and industrial, have varying requirements and usage patterns. Additionally, many utility companies implement time-dependent charges to encourage off-peak usage and optimize resource allocation. However, calculating these tariffs manually can be complex and error-prone.

The Electricity Tariff Calculator addresses this challenge by automating the tariff calculation process. It is a valuable tool for consumers to estimate their electricity costs accurately and for utility companies to ensure fair billing practices.

2.2 Objectives

The primary objectives of the Electricity Tariff Calculator are as follows:
- Develop a user-friendly program for calculating electricity tariffs.
- Differentiate tariffs between domestic and industrial consumers.
- Apply time-dependent charges based on usage hours.
- Facilitate easy data input and transparent tariff display.

3. Technologies Used

The project utilizes the following technologies:
- **C Programming Language:** The core logic of the calculator is implemented in the C programming language.
- **Standard C Libraries:** The `<stdio.h>` and `<stdlib.h>` libraries are used for input/output operations and memory allocation.

4. System Architecture

4.1 Front-End

The front-end of the Electricity Tariff Calculator is implemented as a command-line interface (CLI). It provides a text-based user interface for inputting consumer data and viewing calculated tariffs. The CLI is designed to be intuitive and user-friendly, guiding users through the input process.

4.2 Back-End

The back-end of the system comprises a set of C functions responsible for tariff calculation, memory management, and data processing. It takes user input, performs the necessary calculations, and generates tariff information for display.

4.3 Database

As this is a lightweight command-line utility, it does not require a database for data storage. Consumer data is temporarily stored in memory during program execution.

5. Project Modules

5.1 Module 1: Electricity Tariff Calculation

Purpose: This module contains functions responsible for calculating electricity tariffs based on consumer type and usage hours.

Functionality: It evaluates the type of consumer, whether domestic or industrial, and applies appropriate tariff rates. Additionally, it considers the time of day to determine whether additional charges apply.

5.2 Module 2: User Input and Data Management

Purpose: Module 2 manages user interactions, data input, and storage.

Functionality: It provides a user-friendly interface for users to input consumer data, including name, type, usage hours, and energy consumption. It also manages data storage during program execution.

6. Design and Implementation

6.1 Front-End Design

The front-end design focuses on creating a straightforward and interactive command-line interface. It utilizes C's `printf` and `scanf` functions for input and output. Users are guided through the process of entering consumer data, and tariff information is displayed clearly.

6.2 Back-End Design

The back-end is designed to be modular and efficient. It includes functions for tariff calculation, data validation, and memory management. The `calculateTariff` function, in particular, performs the core tariff calculations based on the provided inputs.

6.3 Database Design

Given the simplicity of the application, no database design is necessary. Consumer data is stored in memory while the program is running, ensuring efficient data access.

7. Features and Functionality

7.1 Feature 1: Input Consumer Data

Functionality: Users can input consumer data, including the consumer's name, type (domestic or industrial), usage hours, and energy consumption.

Importance: This feature ensures that the tariff calculation is tailored to each consumer's specific characteristics.

7.2 Feature 2: Tariff Calculation

Functionality: The application calculates electricity tariffs based on the consumer type and usage hours. It differentiates between domestic and industrial tariffs and applies time-dependent charges when applicable.

Importance: Accurate tariff calculation ensures transparency and fairness in billing, promoting responsible electricity consumption.

7.3 Feature 3: Display Tariffs

Functionality: The calculated tariffs are displayed to the user in a clear and understandable format. This includes showing the consumer's name, type, and the calculated tariff amount.

Importance: Transparent tariff presentation empowers consumers to make informed decisions regarding their electricity usage.

8. Testing

8.1 Unit Testing

Unit tests were conducted to verify the correctness of individual functions, including the `calculateTariff` function. Test cases covered various consumer types and usage scenarios.

8.2 Integration Testing

Integration tests ensured that different modules of the program work together seamlessly. This included testing data input, tariff calculation, and display.

8.3 User Acceptance Testing

Users were invited to participate in testing the application. Their feedback helped identify usability issues and ensure that the software meets their requirements.

9. Challenges Faced

During the development of the Electricity Tariff Calculator, several challenges were encountered:
- Handling different consumer types and their corresponding tariff rates.
- Implementing time-dependent charges based on usage hours.
- Ensuring the user interface is intuitive and easy to use.
These challenges were overcome through careful design and thorough testing.

10. Future Enhancements

To enhance the Electricity Tariff Calculator in the future, the following improvements could be considered:
- Support for additional consumer types (e.g., commercial).
- Implementation of a graphical user interface (GUI) for improved user experience.
- Integration with external data sources for real-time tariff updates.

11. Conclusion

The Electricity Tariff Calculator is a valuable tool for consumers and utility companies. It simplifies the complex task of calculating electricity tariffs, promotes transparent billing practices, and empowers consumers to make informed decisions about their energy usage. With further enhancements, it has the potential to become an even more valuable utility.

12. References

To study the provided C code for calculating electricity tariffs, you can refer to the following three websites that offer tutorials, examples, and explanations of C programming concepts:

1. GeeksforGeeks - C Programming Language
- GeeksforGeeks is a reputable platform for programming tutorials. Their C Programming section provides comprehensive coverage of C programming concepts with code examples and explanations.
- **Website:** [GeeksforGeeks C Programming](https://www.geeksforgeeks.org/c-programming-language/)

2. Programiz - Learn C Programming
- Programiz offers a beginner-friendly approach to learning C programming. They provide tutorials with code examples and exercises to practice C programming concepts.
- **Website:** [Programiz C Programming](https://www.programiz.com/c-programming)

3. Learn-C.org - C Tutorial
- Learn-C.org provides interactive C programming tutorials with examples and exercises. It's a useful resource for beginners looking to understand C programming concepts step by step.
- Website: [Learn-C.org](https://www.learn-c.org/)

13. Appendices

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

// Define a structure to store consumer information
struct Consumer {
    char name[50];
    char type;      // 'D' for domestic, 'I' for industrial
    int time;       // Time at which load is required (in hours)
    double energy;  // Amount of energy utilized (in kWh)
    double tariff;  // Calculated tariff
};

// Function to calculate the electricity tariff
void calculateTariff(struct Consumer *consumer) {
    consumer->tariff = 0.0;

    // Calculate based on the type of consumer
    if (consumer->type == 'D') {
        consumer->tariff = 0.1 * consumer->energy; // Domestic consumers pay $0.1 per kWh
    } else if (consumer->type == 'I') {
        consumer->tariff = 0.15 * consumer->energy; // Industrial consumers pay $0.15 per kWh
    } else {
        printf("Invalid consumer type for %s.\n", consumer->name);
    }

    // Apply time-dependent charges
    if (consumer->time >= 9 && consumer->time <= 17) {
        consumer->tariff += 0.05 * consumer->energy; // Additional daytime charge
    } else {
        consumer->tariff += 0.03 * consumer->energy; // Lower nighttime charge
    }
}

int main() {
    int numConsumers;

    // Read the number of consumers from the user
```

```
    printf("Enter the number of consumers: ");
    scanf("%d", &numConsumers);

    // Allocate memory for an array of Consumer structures
    struct Consumer *consumers = (struct Consumer *)malloc(numConsumers *
sizeof(struct Consumer));
    if (consumers == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }

    // Read consumer data from the user
    for (int i = 0; i < numConsumers; i++) {
        printf("Enter consumer %d information:\n", i + 1);
        printf("Name: ");
        scanf("%s", consumers[i].name);
        printf("Type (D for domestic, I for industrial): ");
        scanf(" %c", &consumers[i].type); // Note the space before %c to consume newline
characters
        printf("Time (in hours): ");
        scanf("%d", &consumers[i].time);
        printf("Energy consumed (in kWh): ");
        scanf("%lf", &consumers[i].energy);

        // Calculate the tariff for the consumer
        calculateTariff(&consumers[i]);
    }

    // Display the calculated tariffs for each consumer
    printf("\nElectricity Tariffs:\n");
    for (int i = 0; i < numConsumers; i++) {
        printf("Name: %s, Type: %c, Tariff: $%.2lf\n", consumers[i].name,
consumers[i].type, consumers[i].tariff);
    }

    // Free dynamically allocated memory
    free(consumers);

    return 0;
}
```

13.1 Screenshots

1.Domestic:

```
Enter the number of consumers: 1
Enter consumer 1 information:
Name: Akshay
Type (D for domestic, I for industrial): D
Time (in hours): 3
Energy consumed (in kWh): 150

Electricity Tariffs:
Name: Akshay, Type: D, Tariff: $19.50
```

2. Industrial:

```
Enter the number of consumers: 1
Enter consumer 1 information:
Name: JoystonMonthero
Type (D for domestic, I for industrial): I
Time (in hours): 10
Energy consumed (in kWh): 400

Electricity Tariffs:
Name: JoystonMonthero, Type: I, Tariff: $80.00
```

13.2 Code Snippets

1. "calculateTariff" Function:

```c
// Function to calculate the electricity tariff
void calculateTariff(struct Consumer *consumer) {
    consumer->tariff = 0.0;

    // Calculate based on the type of consumer
    if (consumer->type == 'D') {
        consumer->tariff = 0.1 * consumer->energy; // Domestic consumers pay $0.1 per kWh
    } else if (consumer->type == 'I') {
        consumer->tariff = 0.15 * consumer->energy; // Industrial consumers pay $0.15 per kWh
    } else {
        printf("Invalid consumer type for %s.\n", consumer->name);
    }

    // Apply time-dependent charges
    if (consumer->time >= 9 && consumer->time <= 17) {
        consumer->tariff += 0.05 * consumer->energy; // Additional daytime charge
    } else {
        consumer->tariff += 0.03 * consumer->energy; // Lower nighttime charge
    }
}
```

2. "main" Function:

```c
int main() {
    int numConsumers;
    // Read the number of consumers from the user
    printf("Enter the number of consumers: ");
    scanf("%d", &numConsumers);
    // Allocate memory for an array of Consumer structures
    struct Consumer *consumers = (struct Consumer *)malloc(numConsumers * sizeof(struct Consumer));
    if (consumers == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }
    // Read consumer data from the user
    for (int i = 0; i < numConsumers; i++) {
        printf("Enter consumer %d information:\n", i + 1);
        printf("Name: ");
        scanf("%s", consumers[i].name);
        printf("Type (D for domestic, I for industrial): ");
        scanf(" %c", &consumers[i].type); // Note the space before %c to consume newline characters
        printf("Time (in hours): ");
        scanf("%d", &consumers[i].time);
        printf("Energy consumed (in kWh): ");
        scanf("%lf", &consumers[i].energy);

        // Calculate the tariff for the consumer
        calculateTariff(&consumers[i]);
    }
    // Display the calculated tariffs for each consumer
    printf("\nElectricity Tariffs:\n");
    for (int i = 0; i < numConsumers; i++) {
        printf("Name: %s, Type: %c, Tariff: $%.2lf\n", consumers[i].name, consumers[i].type, consumers[i].tariff);
    }

    // Free dynamically allocated memory
    free(consumers);

    return 0;
}
```