Project Report: Electricity Billing System

Abstract

The Electricity Billing System is a software application designed to simplify the management of electricity bills. It offers users the ability to record, calculate, and manipulate electricity bill information in an efficient and user-friendly manner. This project report outlines the system's objectives, design, implementation, and potential for future enhancements.

The primary goals of the system are to provide users with a platform for maintaining detailed records of their electricity bills, automate the calculation of bill amounts, ensure user-friendliness through an intuitive interface, and uphold data security. Achieving these objectives enhances the user's ability to manage their electricity expenses effectively.

The system is built around a structured data model using the `Bill` struct, which encapsulates critical bill details. It offers three main functions: adding a bill, displaying all bills, and deleting a bill. The total bill amount is calculated using a predefined electricity rate.

Future enhancements could include user authentication, data persistence through file I/O, advanced search and filtering capabilities, bill summaries, data backup, and a comprehensive billing history log. These improvements would make the system more versatile and suitable for broader usage.

In conclusion, the Electricity Billing System project provides a solid foundation for efficient electricity bill management. Continuous development and user feedback will be pivotal in adapting the system to meet evolving requirements and ensure a seamless experience for users.

Introduction

The **Electricity Billing System** project aims to develop a user-friendly software application for managing and recording electricity bills. The system is designed to provide users with the ability to add, view, and delete electricity bills. This project report outlines the objectives, design, implementation, and future enhancements of the system.

Objectives

The primary objectives of the Electricity Billing System are as follows:

1. **Record Keeping**: To provide a digital platform for users to maintain records of their electricity bills, including the date, units consumed, and total bill amount.

2. **Efficient Bill Calculation**: To automate the calculation of the total bill amount based on the number of units consumed, using a predefined electricity rate.

3. **User-Friendly Interface**: To offer an intuitive and user-friendly interface for easy data entry and retrieval of bill information.

4. **Data Security**: To ensure the security and privacy of user data, implementing proper data storage and access control mechanisms.

System Design

Data Structure

The core data structure used in this system is the `Bill` struct, which contains the following fields:

- `id`: A unique identifier for each bill.

- `day`, `month`, `year`: Fields to store the date of the bill.

- `unitsConsumed`: A double to store the number of units of electricity consumed.

- `totalBill`: A double to store the total bill amount.

Functionality

The system provides the following functionalities:

1. **Adding a Bill**:

   - Users can add a new bill to the system.

   - The system prompts the user for the date and units consumed.

   - It calculates the total bill amount using a predefined electricity rate and assigns a unique ID to the bill.

   - The bill is stored in memory.

2. **Displaying Bills**:

   - Users can view all the bills stored in the system.

- The system displays the bills in a tabular format, including their IDs, dates, units consumed, and total bill amounts.

3. **Deleting a Bill**:

   - Users can delete a bill by specifying its unique ID.

   - The system searches for the bill with the given ID, deletes it, and updates the bill count.

Algorithm: Electricity Billing System

```
Step 1: Initialize an array of Bill structures (bills) to store electricity bill records.
Step 2: Initialize a variable billCount to keep track of the number of bills (initially set to 0).
Step 3: Define a constant ELECTRICITY_RATE for the electricity cost per unit (e.g., 0.10 per unit).

Step 4: Display "Electricity Billing System Menu:" with options:
    1. Add a new bill
    2. Display all bills
    3. Delete a bill
    4. Exit

Step 5: Read the user's choice (choice) from the keyboard.

Step 6: Switch on choice:
  Case 1:
    1.1: Check if billCount < 100 (to ensure there is space for a new bill).
    1.2: If true, do the following:
      1.2.1: Prompt the user for the date (day, month, year) and units consumed.
      1.2.2: Calculate the total bill using unitsConsumed * ELECTRICITY_RATE.
      1.2.3: Generate a unique ID for the new bill (e.g., increment billCount).
      1.2.4: Create a new Bill structure with the entered data and unique ID.
      1.2.5: Store the new Bill structure in the bills array at index billCount.
      1.2.6: Increment billCount.
      1.2.7: Display "Bill added successfully."
    1.3: If billCount >= 100, display "Maximum number of bills reached."

  Case 2:
    2.1: Display "Bills:"
    2.2: Loop through the bills array from 0 to billCount - 1:
      2.2.1: Display the bill's ID, date (formatted), units consumed, and total bill amount.
    2.3: If billCount is 0, display "No bills found."

  Case 3:
    3.1: Prompt the user to enter the ID of the bill to delete (idToDelete).
    3.2: Initialize a variable found to 0.
    3.3: Loop through the bills array from 0 to billCount - 1:
      3.3.1: If bills[i].id equals idToDelete:
        3.3.1.1: Set found to 1.
        3.3.1.2: Delete the bill at index i by shifting remaining bills.
        3.3.1.3: Decrement billCount.
        3.3.1.4: Display "Bill with ID X deleted successfully."
        3.3.1.5: Break out of the loop.
    3.4: If found is still 0, display "Bill with ID X not found."

  Case 4:
    4.1: Display "Exiting the program."
    4.2: Exit the program.

  Default:
    Display "Invalid choice. Please try again."

Step 7: Repeat from Step 4 until choice is 4 (Exit).

Step 8: End of the program.
```

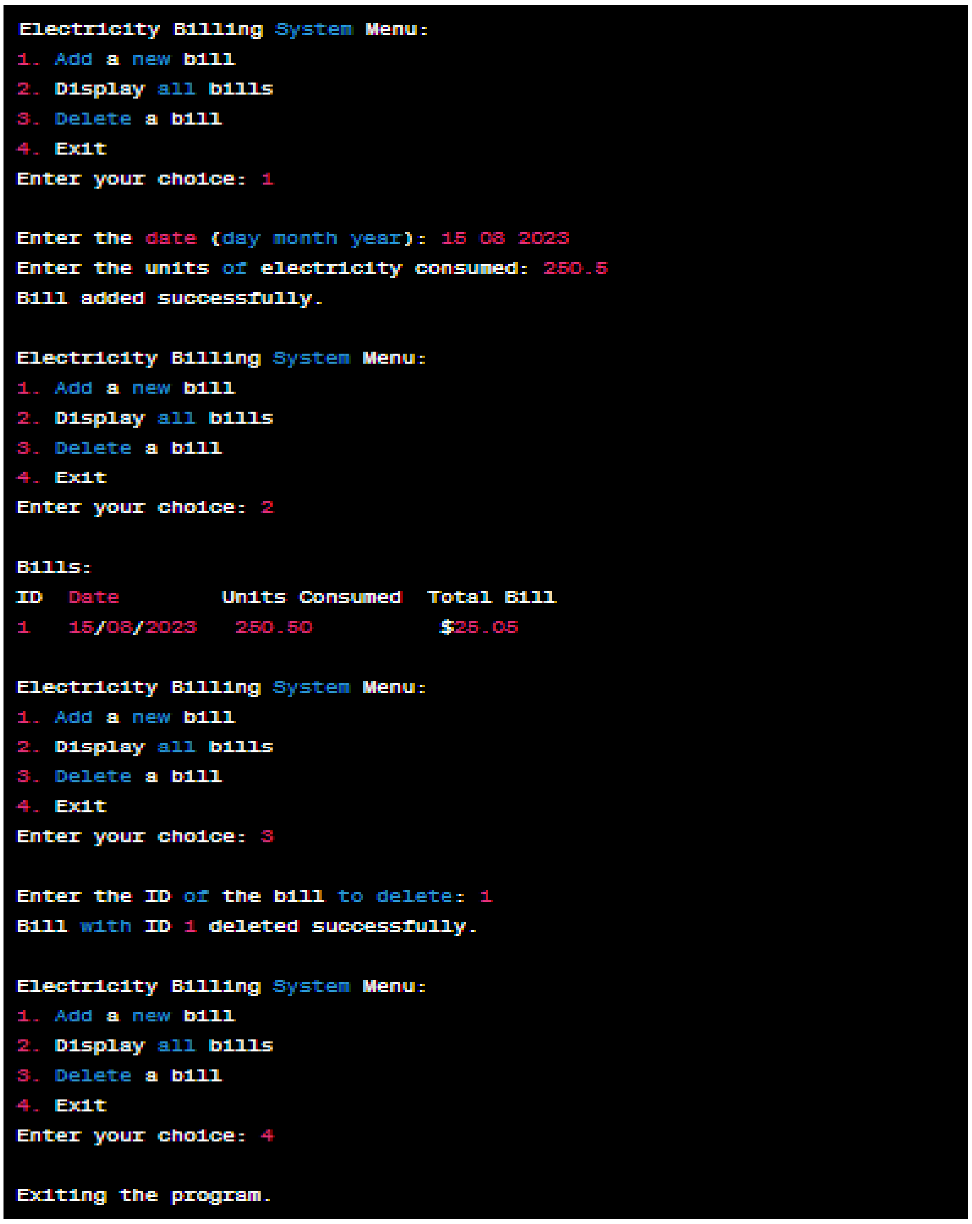
SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
struct Bill {
    int id;
    int day;
    int month;
    int year;
    double unitsConsumed;
    double totalBill;
};
double calculateBill(double units) {
    #define ELECTRICITY_RATE 0.10
    return units * ELECTRICITY_RATE;
}
void addBill(struct Bill bills[], int *billCount) {
    if (*billCount < 100) { // Assuming a maximum of 100 bills
        struct Bill newBill;
        printf("Enter the date (day month year): ");
        scanf("%d %d %d", &newBill.day, &newBill.month, &newBill.year);
        printf("Enter the units of electricity consumed: ");
        scanf("%lf", &newBill.unitsConsumed);
        newBill.totalBill = calculateBill(newBill.unitsConsumed);
        newBill.id = *billCount + 1;
        bills[*billCount] = newBill;
        (*billCount)++;
        printf("Bill added successfully.\n");
    } else {
        printf("Maximum number of bills reached.\n");
    }
}
void displayBills(struct Bill bills[], int billCount) {
    printf("Bills:\n");
    printf("ID  Date       Units Consumed  Total Bill\n");
    for (int i = 0; i < billCount; i++) {
        printf("%-3d %02d/%02d/%04d   %.2lf           $%.2lf\n", bills[i].id, bills[i].day, bills[i].month,
bills[i].year, bills[i].unitsConsumed, bills[i].totalBill);
    }
}
void deleteBill(struct Bill bills[], int *billCount, int idToDelete) {
    int found = 0;
    for (int i = 0; i < *billCount; i++) {
        if (bills[i].id == idToDelete) {
            found = 1;
            // Shift remaining bills to fill the gap
            for (int j = i; j < *billCount - 1; j++) {
                bills[j] = bills[j + 1];
            }
            (*billCount)--;
            printf("Bill with ID %d deleted successfully.\n", idToDelete);
            break;
        }
    }
    if (!found) {
        printf("Bill with ID %d not found.\n", idToDelete);
    }
}
int main() {
    struct Bill bills[100]; // Array to store bill records
```

```
  int billCount = 0;
int choice;
  do {
      printf("\nElectricity Billing System Menu:\n");
      printf("1. Add a new bill\n");
      printf("2. Display all bills\n");
      printf("3. Delete a bill\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
    switch (choice) {
        case 1:
            addBill(bills, &billCount);
            break;
        case 2:
            displayBills(bills, billCount);
            break;
        case 3:
            {
                int idToDelete;
                printf("Enter the ID of the bill to delete: ");
                scanf("%d", &idToDelete);
                deleteBill(bills, &billCount, idToDelete);
            }
            break;
        case 4:
            printf("Exiting the program.\n");
            break;
        default:     printf("Invalid choice. Please try again.\n");
            break;
      }
    } while (choice != 4);
return 0;
}
```

Output

The output of the Electricity Billing System appears in the console or terminal window. Below is an example of the program's output during typical usage:



```
Electricity Billing System Menu:
1. Add a new bill
2. Display all bills
3. Delete a bill
4. Exit
Enter your choice: 1

Enter the date (day month year): 15 08 2023
Enter the units of electricity consumed: 250.5
Bill added successfully.

Electricity Billing System Menu:
1. Add a new bill
2. Display all bills
3. Delete a bill
4. Exit
Enter your choice: 2

Bills:
ID  Date          Units Consumed  Total Bill
1   15/08/2023    250.50           $25.05

Electricity Billing System Menu:
1. Add a new bill
2. Display all bills
3. Delete a bill
4. Exit
Enter your choice: 3

Enter the ID of the bill to delete: 1
Bill with ID 1 deleted successfully.

Electricity Billing System Menu:
1. Add a new bill
2. Display all bills
3. Delete a bill
4. Exit
Enter your choice: 4

Exiting the program.
```

The above example demonstrates a typical interaction with the system:

1. The user adds a new bill with a specific date and units consumed.

2. They then choose to display all bills, which shows the added bill with its ID, date, units consumed, and total bill amount.

3. The user proceeds to delete the bill by specifying its ID.

4. Finally, they exit the program.

The program provides clear prompts and feedback to guide the user through each step and performs the requested operations accurately.

Implementation

Data Storage

The system uses an array of `Bill` structs to store bill records. The maximum number of bills is set to 100 in this implementation. However, this can be adjusted as needed.

Calculating the Total Bill

The total bill amount is calculated using the `calculateBill` function, which takes the number of units consumed as input and multiplies it by a predefined electricity rate (0.10 in this case).

User Interface

The user interface is implemented using a `do-while` loop, presenting users with a menu of options. Users can choose to add a new bill, display all bills, delete a bill, or exit the program. User input is validated to handle invalid choices.

Future Enhancements

To enhance the functionality and usability of the Electricity Billing System, the following features can be added in future iterations:

1. **User Authentication**: Implement user authentication to ensure that only authorized users can access and modify bill records.

2**. Data Persistence**: Add file I/O operations to store bill records in a file for persistent storage. This ensures that data is not lost when the program is closed.

3. **Search and Filter**: Allow users to search for specific bills by date range, ID, or other criteria, making it easier to locate and view specific records.

4. **Bill Summaries**: Provide summarized billing information, such as monthly or yearly totals, to help users analyze their electricity usage and expenses.

5**. Data Backup**: Implement a backup and restore feature to prevent data loss in case of system failures or accidental deletion of bills.

6. **Billing History**: Maintain a history of bill changes and updates, including the date and user responsible for the change.

Conclusion

The Electricity Billing System project has successfully developed a basic system for managing electricity bills. It allows users to add, view, and delete bill records efficiently. While this version of the system serves as a foundation, there is ample room for enhancement and expansion.

Future iterations of the system can include features such as user authentication, data persistence, advanced search and filter options, bill summaries, data backup, and a comprehensive billing history log. These additions will make the system more robust, user-friendly, and suitable for real-world use.

The Electricity Billing System project demonstrates the potential for using software to simplify and streamline everyday tasks, such as managing household bills. With continuous improvement and user feedback, this system can evolve to meet the evolving needs of its users.