

INTRODUCTION

ELECTRICITY BILL GENERATOR:

Electricity Bill Generator is a short program designed to calculate the bill and then display it. The Electricity Bill Generator is a concise program that efficiently computes and generates electricity bills by processing user input, ensuring accuracy and ease of use. It serves a user-friendly solution for transparent and precise billing.

OBJECTIVE:

The final bill is presented in a well-formatted manner, offering a clear overview of the customer's billing details. The output includes essential information such as customer name, ID, address, units consumed, tax amount, total bill, and grand total bill.

WHAT MY PROGRAM DOES:

This program performs the following actions:

- ✓ Calculation of Total Bill
- ✓ Customer Information Gathering
- ✓ Handling of Previous Unpaid Bills
- ✓ Tax Calculation and Grand Total
- ✓ Display of Final Bill

Calculation of Total Bill:

Calculating the entire bill depending on the amount of electricity used. After the user enters consumed data, the computer calculates the overall bill amount by applying a fixed rate per unit.

Customer Information Gathering:

The program includes a function to gather essential customer information, such as customer ID, name, address, and the current date. This ensures that the generated bills are associated with the correct customer details.

Handling of Previous Unpaid Bills:

The program provides a solution for dealing with previous unpaid bills. It prompts users to input information about outstanding payments, calculates fines if needed, and issues a warning for an excessive number of unpaid months, ensuring a user-friendly approach to handling overdue bills.

Tax Calculation and Grand Total:

To mirror real-world billing situations, the program includes a tax calculation feature using a fixed rate. The grand total bill is accurately computed by adding the calculated tax amount to the initial total bill, contributing to a comprehensive and realistic billing representation.

Display of Final Bill:

The final bill is presented in a well-formatted manner, offering a clear overview of the customer's billing details. The output includes essential information such as customer name, ID, address, units consumed, tax amount, total bill, and grand total bill.

PROGRAM STRUCTURE

HEADER FILES:

The program includes the header files such as

```
#include "stdafx.h"
```

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

```
#include <string.h>
```

STRUCTURES:

struct customer:

This structure represents a customer and has the following fields:

name: A character array to store the customer's name (up to 50 characters).

id: An integer to store the customer's ID.

units: An integer to store the number of units consumed.

address: A character array to store the customer's address (up to 50 characters).

date: A character array to store the date of the bill (up to 15 characters).

struct bill:

This structure represents a bill and has the following fields:

id: An integer to store the customer's ID.

total_bill: An integer to store the total bill amount.

fined_months: An integer to store the number of months for which the customer has unpaid bills.

tax_amount: A float to store the calculated tax amount.

g_total_bill: A float to store the grand total bill (total bill + tax).

FUNCTIONS:

calculate_total_bill():

- Asks the user to input the number of units consumed.
- Calculates the total bill by multiplying the number of units by a fixed price per unit .

get_customer_data():

- Gathers information about the customer.
- Prompts the user to input the customer's ID, name, address, and the current date.

check_previous_bills():

- Asks the user if they have paid the previous bills (Y/N).
- If the response is 'N' or 'n' (indicating unpaid bills), the program:
 - Prompts the user to input the number of months with unpaid bills.
 - For each month, asks the user to input the unpaid amount.
 - Issues a warning if the number of unpaid months is greater than 3.
 - Calculates fines (1000 per unpaid month) and adds them to the total bill.

add_tax():

- Calculates the tax amount by applying a fixed tax rate of 5% to the total bill.
- Updates the grand total bill by adding the calculated tax amount.

display_bill():

Prints the final electricity bill with detailed information:

- Customer Name, ID, and Address.
- Units Consumed.
- Issue Date.
- Tax Amount.
- Total Bill Amount.
- Grand Total Bill Amount.

FILE HANDLING:

Made a pointer file name bill to store data. In this function we use `fopen_s(bill)` to open the file, `fprintf` to write and `fclose(bill)` to close the file. Through filing we can keep the previous record. The program uses file handling to write the generated bill information to a text file ("bill.txt").

- **FILE* bill:**
Declares a file pointer named bill. This pointer is used to reference the file stream.
- **fopen_s:**
Opens a file named "bill.txt" in append mode ("a"). It opens the file and associates it with the file pointer bill.
- **fprintf:**
The **fprintf** function in C is used to write formatted data to a file. It is similar to **printf**, but instead of printing to the console, it writes the formatted output to a specified file.
- **fclose:**
Closes the file associated with the file pointer **bill**. Closing the file is important to ensure that changes are saved, and system resources are released.

main() FUNCTION:

The **main()** function serves as the central control hub for the Electricity Bill Generator program. It directs the flow of execution, orchestrating key functions to ensure a smooth billing process. Here's a breakdown of its role in the program:

calculate_total_bill():

Invokes the function to calculate the total electricity bill based on the user-inputted units consumed.

get_customer_data():

Calls the function to gather essential customer information, including ID, name, address, and date.

check_previous_bills():

Executes the function to determine if there are any outstanding payments from previous months. If so, it prompts the user to input the details and calculates fines accordingly.

add_tax():

Invokes the function to calculate the tax amount based on a fixed tax rate and computes the grand total bill by adding the tax to the initial total bill.

display_bill():

Calls the function to present a well-formatted summary of the electricity bill, including customer details, units consumed, tax amount, total bill, and grand total bill.

RETURN STATEMENT:

Concludes the **main()** function by returning 0, indicating a successful execution.

ADVANTAGES

- **User-Friendly Input:**

The program uses **scanf_s** to take user input, making it more secure by specifying the buffer size. This helps prevent buffer overflow vulnerabilities.

- **Modular Design:**

The program is organized into functions, each responsible for a specific task (e.g., calculating the total bill, getting customer data). This modular design enhances code readability and maintainability.

- **Structured Data Storage:**

The use of structures (**struct customer** and **struct bill**) provides a clean and organized way to store related data, making it easier to manage and access customer and billing information.

- **Detailed Customer Information:**

The program captures essential customer details, such as name, ID, address, and units consumed. This allows for a comprehensive and informative electricity bill.

- **Handling Unpaid Bills:**

The program checks for unpaid bills from previous months, prompting the user to input the details. It calculates fines and issues a warning if the number of unpaid months exceeds a threshold, providing a mechanism for handling overdue payments.

- **Tax Calculation:**

The program includes a function to calculate tax based on a fixed rate, enhancing the realism of the billing process.

- **Formatted Output:**

The final bill is displayed in a well-formatted manner, improving user readability and understanding. The bill output includes relevant customer details, making it easy for the user to review and comprehend.

- **Clear Warning Messages:**

The program issues a warning if the number of unpaid months exceeds a certain limit. This proactive approach informs the user about the potential consequences of not paying overdue bills.

- **Logical Flow:**

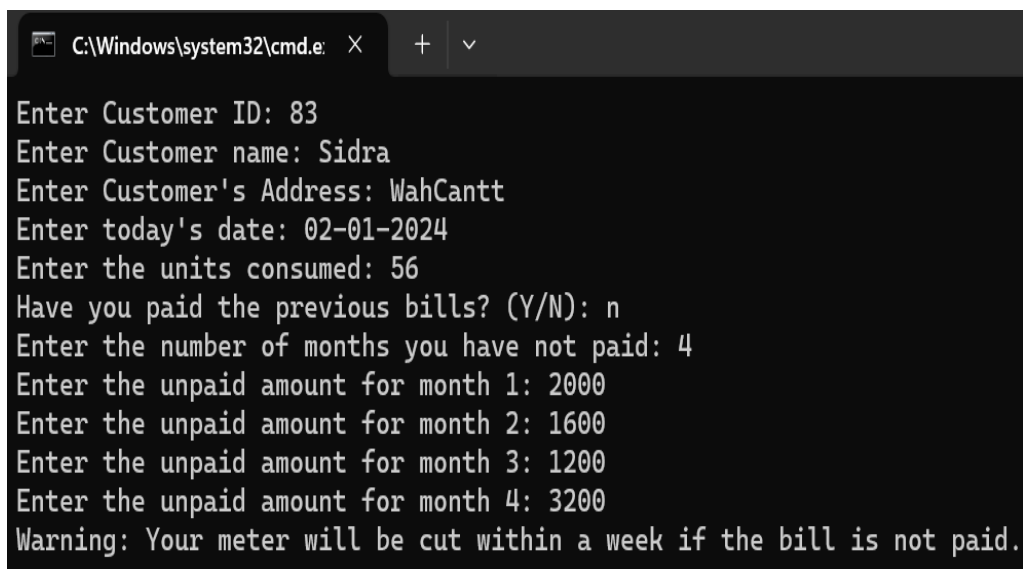
The main function orchestrates the flow of the program in a logical sequence, calling functions in the order necessary for the billing process. This improves the overall structure of the code.

- **Potential for Extension:**

The code is designed in a way that allows for potential extensions, such as supporting multiple customers or bills, and it could be adapted for more complex scenarios.

OUTPUT:

The output of the program is a well-formatted display of the electricity bill on the console, as well as the generation of a text file named "bill.txt" containing the same information.



```
C:\Windows\system32\cmd.e  X  +  v

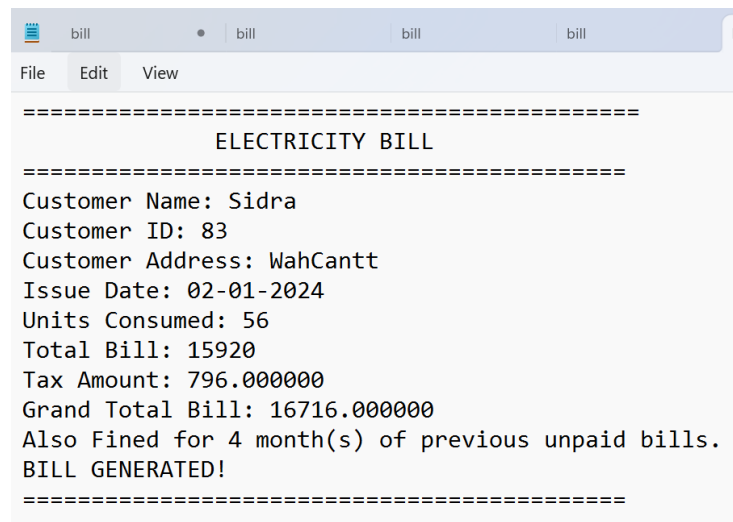
Enter Customer ID: 83
Enter Customer name: Sidra
Enter Customer's Address: WahCantt
Enter today's date: 02-01-2024
Enter the units consumed: 56
Have you paid the previous bills? (Y/N): n
Enter the number of months you have not paid: 4
Enter the unpaid amount for month 1: 2000
Enter the unpaid amount for month 2: 1600
Enter the unpaid amount for month 3: 1200
Enter the unpaid amount for month 4: 3200
Warning: Your meter will be cut within a week if the bill is not paid.
```

The program also display the bill in the output screen.

```
=====
                        ELECTRICITY BILL
=====
Customer Name:      Sidra
Customer ID:        83
Customer Address:   WahCantt
Units Consumed:     56
Issue Date:         02-01-2024
Tax Amount:         796.000000
Total Bill:         15920
Grand Total Bill:   16716.000000

BILL GENERATED!
=====
```

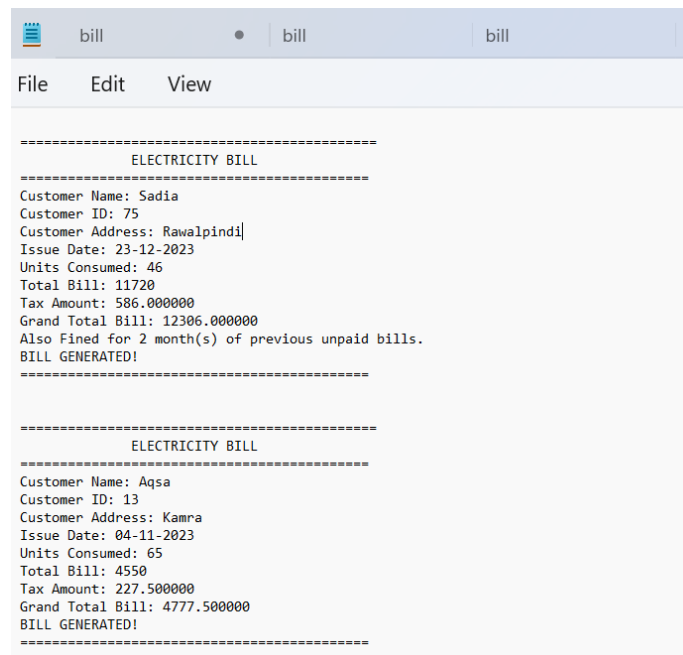
It also display bill in file.



The screenshot shows a file editor window with multiple tabs, one of which is named 'bill'. The editor displays the same electricity bill as shown in the previous image, but with an additional line: 'Also Fined for 4 month(s) of previous unpaid bills.' before the 'BILL GENERATED!' message.

```
=====
                        ELECTRICITY BILL
=====
Customer Name: Sidra
Customer ID: 83
Customer Address: WahCantt
Issue Date: 02-01-2024
Units Consumed: 56
Total Bill: 15920
Tax Amount: 796.000000
Grand Total Bill: 16716.000000
Also Fined for 4 month(s) of previous unpaid bills.
BILL GENERATED!
=====
```

It also keeps previous record save in files.



The screenshot shows a file editor window with multiple tabs, one of which is named 'bill'. The editor displays two electricity bills, one above the other, separated by a blank line. The first bill is for Sadia (Customer ID: 75) and the second is for Aqsa (Customer ID: 13). Both bills include the same fields as the previous ones, but with their respective values.

```
=====
                        ELECTRICITY BILL
=====
Customer Name: Sadia
Customer ID: 75
Customer Address: Rawalpindi
Issue Date: 23-12-2023
Units Consumed: 46
Total Bill: 11720
Tax Amount: 586.000000
Grand Total Bill: 12306.000000
Also Fined for 2 month(s) of previous unpaid bills.
BILL GENERATED!
=====

=====
                        ELECTRICITY BILL
=====
Customer Name: Aqsa
Customer ID: 13
Customer Address: Kamra
Issue Date: 04-11-2023
Units Consumed: 65
Total Bill: 4550
Tax Amount: 227.500000
Grand Total Bill: 4777.500000
BILL GENERATED!
=====
```