NAME: HIRA ASLAM , HANZLA BASHIR

SAP-ID: 59631 , 60059

PRESENTED TO: DR . ASIF NAUMAN

CLASS: BS-IT-1

SUBJECT: PROGRAMING FUNDAMENTALS

DOCUMENTATION

Contents

* [SUPER MARKET BILLING SYSTEM 2](#_Toc168056401)
* [LIBRARIES: 2](#_Toc168056402)
* [INTIALIZATION DATA TYPES: 2](#_Toc168056403)
* [FUNCTIONS: 2](#_Toc168056404)
* [SWITCH STATEMENTS: 2](#_Toc168056405)
* [FUNCTION TO DISPLAY MENU: 3](#_Toc168056406)
* [FUNCTION TO GET PRODUCTS: 3](#_Toc168056407)
* [FUNCTION TO DISPLAY BILL: 4](#_Toc168056408)
* [FUNCTION TO SAVE BILL TO .TXT FILE: 4](#_Toc168056409)
* [CODE: 4](#_Toc168056410)
* [OUT PUT: 7](#_Toc168056411)
* [FLOW CHART: 8](#_Toc168056412)

# SUPER MARKET BILLING SYSTEM

## LIBRARIES:

#include <iostream>

#include <fstream>

#include <string>

## INTIALIZATION DATA TYPES:

struct Product {

    string name;

    float price;

    float quantity;

};

## FUNCTIONS:

void displayMenu();

void getProducts(Product\* products, int& numItems);

void displayBill(Product\* products, int numItems, float grandTotal);

void writeBillToFile(Product\* products, int numItems, float grandTotal, const string& filename);

## SWITCH STATEMENTS:

 switch (choice) {

            case 1:

                if (numItems < MAX\_ITEMS) {

                    getProducts(products, numItems);

                    grandTotal = 0;

                    for (int i = 0; i < numItems; ++i) {

                        grandTotal += products[i].price \* products[i].quantity;

                    }

                } else {

                    cout << "Maximum items reached!" << endl;

                }

                break;

            case 2:

                displayBill(products, numItems, grandTotal);

                break;

            case 3:

                writeBillToFile(products, numItems, grandTotal, filename);

                break;

            case 4:

                cout << "Exiting program." << endl;

                break;

            default:

                cout << "Invalid choice. Please try again." << endl;

        }

## FUNCTION TO DISPLAY MENU:

void displayMenu() {

    cout << "\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    cout << "1. Add Item" << endl;

    cout << "2. Display Bill" << endl;

    cout << "3. Write Bill to File" << endl;

    cout << "4. Exit" << endl;

}

## FUNCTION TO GET PRODUCTS:

void getProducts(Product\* products, int& numItems) {

    cout << "\nEnter details for item " << (numItems + 1) << ":" << endl;

    cout << "Product name: ";

    cin.ignore();

    getline(cin, products[numItems].name);

    cout << "Price: ";

    cin >> products[numItems].price;

    cout << "Quantity: ";

    cin >> products[numItems].quantity;

    numItems++;

}

## FUNCTION TO DISPLAY BILL:

void displayBill(Product\* products, int numItems, float grandTotal) {

    cout << "\n\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    cout << "  Product Name  | Price  | Quantity | Total  " << endl;

    cout << "----------------------------------------------" << endl;

    for (int i = 0; i < numItems; i++) {

        float total = products[i].price \* products[i].quantity;

        cout << "   " << products[i].name << "     "

             << products[i].price << "     "

             << products[i].quantity << "     "

             << total << endl;

    }

    cout << "\nGrand Total: $" << grandTotal << endl;

}

## FUNCTION TO SAVE BILL TO .TXT FILE:

void writeBillToFile(Product\* products, int numItems, float grandTotal, const string& filename) {

    ofstream outfile(filename);

    if (!outfile.is\_open()) {

        cout << "Error: Could not open file " << filename << endl;

        return;

    }

    outfile << "\n\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    outfile << "  Product Name  | Price  | Quantity | Total  " << endl;

    outfile << "----------------------------------------------" << endl;

    for (int i = 0; i < numItems; i++) {

        float total = products[i].price \* products[i].quantity;

        outfile << "  " << products[i].name << "  "

                << products[i].price << "  "

                << products[i].quantity << "  "

                << total << endl;

    }

## CODE:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int MAX\_ITEMS = 20;

struct Product {

    string name;

    float price;

    float quantity;

};

struct Customer {

    string name;

    string email;

};

struct Order {

    Customer customer;

    Product products[MAX\_ITEMS];

    int numItems;

    float grandTotal;

};

void displayMenu();

void getProducts(Product\* products, int& numItems);

void displayBill(const Order& order);

void writeBillToFile(const Order& order, const string& filename);

bool signIn();

bool login();

void getCustomerDetails(Customer& customer);

int main() {

    if (!login()) {

        cout << "Authentication failed. Exiting program." << endl;

        return 0;

    }

    char choice;

    Order order;

    order.numItems = 0;

    order.grandTotal = 0;

    string filename = "bill.txt";

    getCustomerDetails(order.customer);

    do {

        displayMenu();

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case '1':

                if (order.numItems < MAX\_ITEMS) {

                    getProducts(order.products, order.numItems);

                    order.grandTotal = 0;

                    for (int i = 0; i < order.numItems; ++i) {

                        order.grandTotal += order.products[i].price \* order.products[i].quantity;

                    }

                } else {

                    cout << "Maximum items reached!" << endl;

                }

                break;

            case '2':

                displayBill(order);

                break;

            case '3':

                writeBillToFile(order, filename);

                break;

            case '4':

                cout << "Exiting program." << endl;

                break;

            default:

                cout << "Invalid choice. Please try again." << endl;

        }

    } while (choice != '4');

    return 0;

}

void displayMenu() {

    cout << "\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    cout << "1. Add Item" << endl;

    cout << "2. Display Bill" << endl;

    cout << "3. Write Bill to File" << endl;

    cout << "4. Exit" << endl;

}

void getProducts(Product\* products, int& numItems) {

    cout << "\nEnter details for item " << (numItems + 1) << ":" << endl;

    cout << "Product name: ";

    cin.ignore();

    getline(cin, products[numItems].name);

    cout << "Price: ";

    cin >> products[numItems].price;

    cout << "Quantity: ";

    cin >> products[numItems].quantity;

    numItems++;

}

void displayBill(const Order& order) {

    cout << "\n\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    cout << "Customer Name: " << order.customer.name << endl;

    cout << "Customer Email: " << order.customer.email << endl;

    cout << "----------------------------------------------" << endl;

    cout << "  Product Name  | Price  | Quantity | Total  " << endl;

    cout << "----------------------------------------------" << endl;

    for (int i = 0; i < order.numItems; i++) {

        float total = order.products[i].price \* order.products[i].quantity;

        cout << "   " << order.products[i].name << "     "

             << order.products[i].price << "     "

             << order.products[i].quantity << "     "

             << total << endl;

    }

    cout << "\nGrand Total: $" << order.grandTotal << endl;

}

void writeBillToFile(const Order& order, const string& filename) {

    ofstream outfile(filename);

    if (!outfile.is\_open()) {

        cout << "Error: Could not open file " << filename << endl;

        return;

    }

    outfile << "\n\n\*\*\* SUPERMARKET BILLING SYSTEM \*\*\*\n" << endl;

    outfile << "Customer Name: " << order.customer.name << endl;

    outfile << "Customer Email: " << order.customer.email << endl;

    outfile << "----------------------------------------------" << endl;

    outfile << "  Product Name  | Price  | Quantity | Total  " << endl;

    outfile << "----------------------------------------------" << endl;

    for (int i = 0; i < order.numItems; i++) {

        float total = order.products[i].price \* order.products[i].quantity;

        outfile << "  " << order.products[i].name << "  "

                << order.products[i].price << "  "

                << order.products[i].quantity << "  "

                << total << endl;

    }

    outfile << "\nGrand Total: $" << order.grandTotal << endl;

    outfile.close();

    cout << "Bill successfully written to " << filename << endl;

}

bool signIn() {

    string username, password, confPassword;

    cout << "Sign Up" << endl;

    cout << "Enter username: ";

    cin >> username;

    cout << "Enter password: ";

    cin >> password;

    cout << "Confirm password: ";

    cin >> confPassword;

    if (password != confPassword) {

        cout << "Passwords do not match. Try again." << endl;

        return false;

    }

    ofstream outfile("users.txt", ios::app);

    if (!outfile.is\_open()) {

        cout << "Error: Could not open file to save user data." << endl;

        return false;

    }

    outfile << username << " " << password << endl;

    outfile.close();

    cout << "Sign up successful. You can now log in." << endl;

    return true;

}

bool login() {

    string username, password, storedUsername, storedPassword;

    char choice;

    cout << "Do you have an account? (y/n): ";

    cin >> choice;

    if (choice == 'n' || choice == 'N') {

        if (!signIn()) {

            return false;

        }

    }

    cout << "Log In" << endl;

    cout << "Enter username: ";

    cin >> username;

    cout << "Enter password: ";

    cin >> password;

    ifstream infile("users.txt");

    if (!infile.is\_open()) {

        cout << "Error: Could not open file to read user data." << endl;

        return false;

    }

    while (infile >> storedUsername >> storedPassword) {

        if (username == storedUsername && password == storedPassword) {

            infile.close();

            cout << "Login successful." << endl;

            return true;

        }

    }

    infile.close();

    cout << "Invalid username or password." << endl;

    return false;

}

void getCustomerDetails(Customer& customer) {

    cout << "Enter customer details:" << endl;

    cout << "Customer name: ";

    cin.ignore();

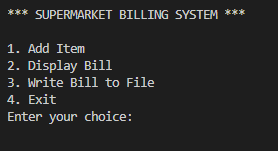
    getline(cin, customer.name);

    cout << "Customer email: ";

    getline(cin, customer.email);

}

## OUT PUT:



## FLOW CHART:

