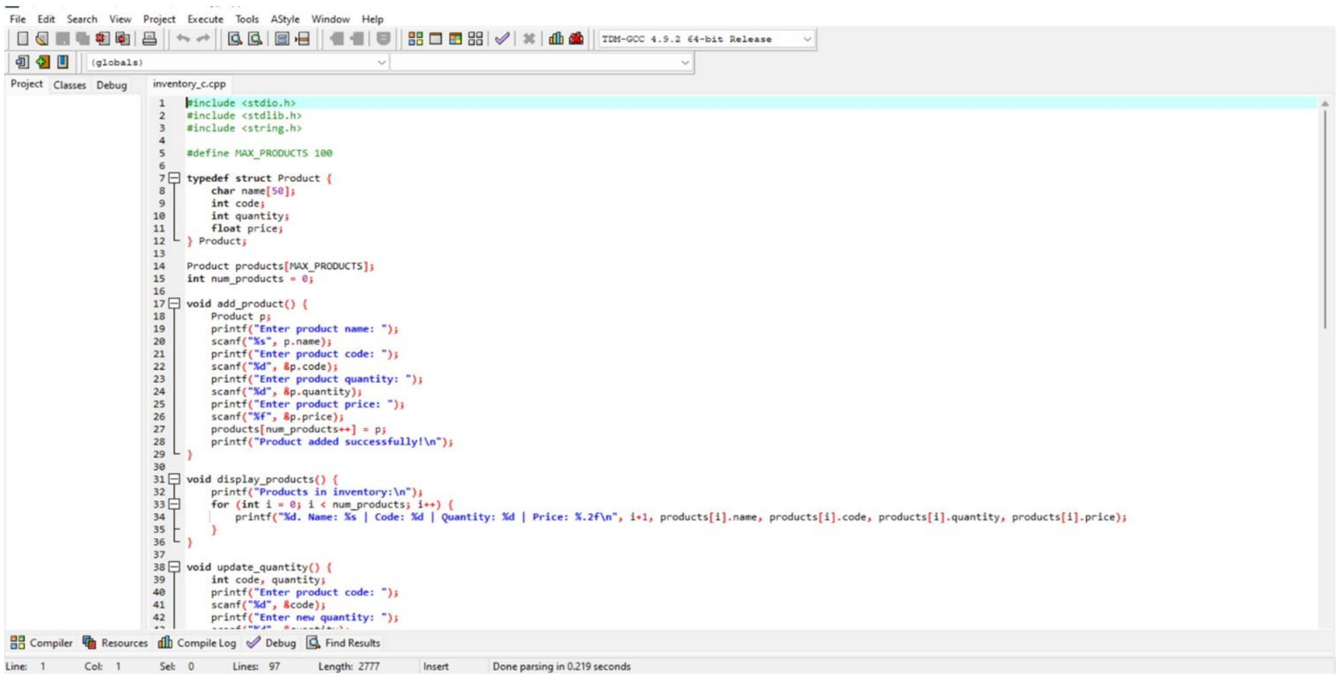
Subject Name : Programming for Problem solving using C

Subject Code : ESC103(Pr.)

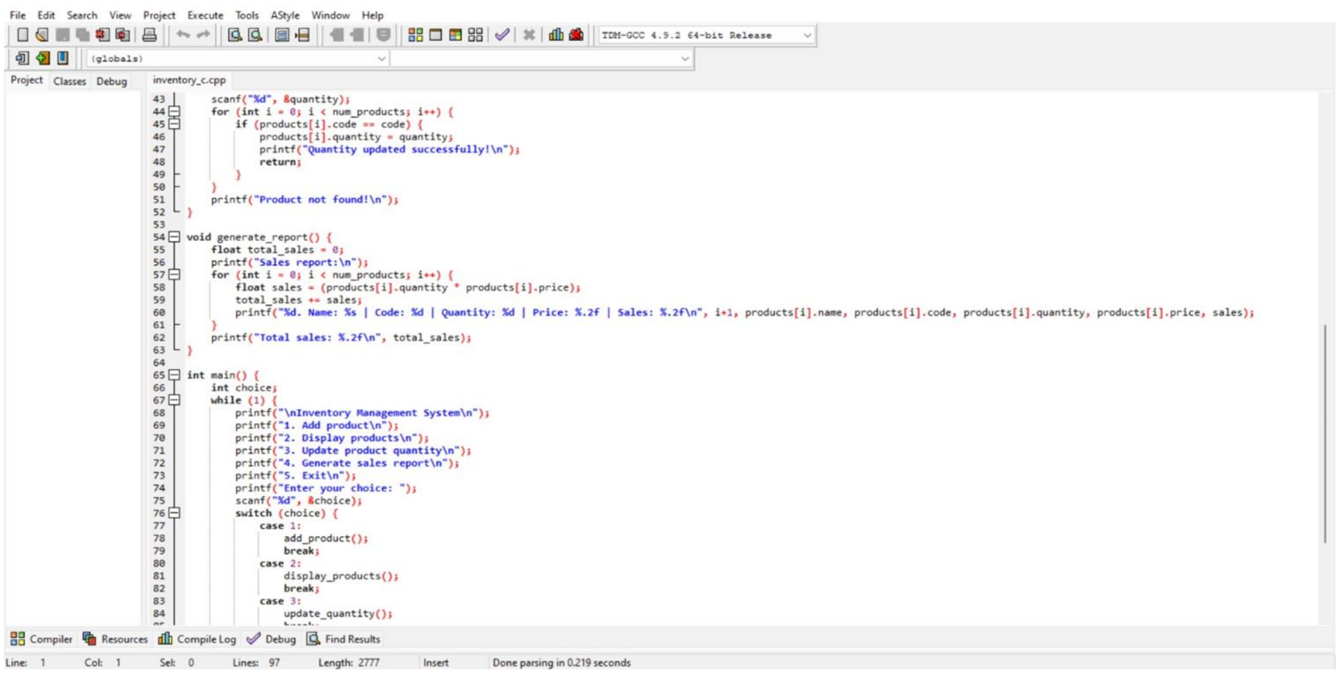
Topic : Inventory management system : Create a program that manages inventory,including tracking product sales, managing stock,and generating reports

Source Code :



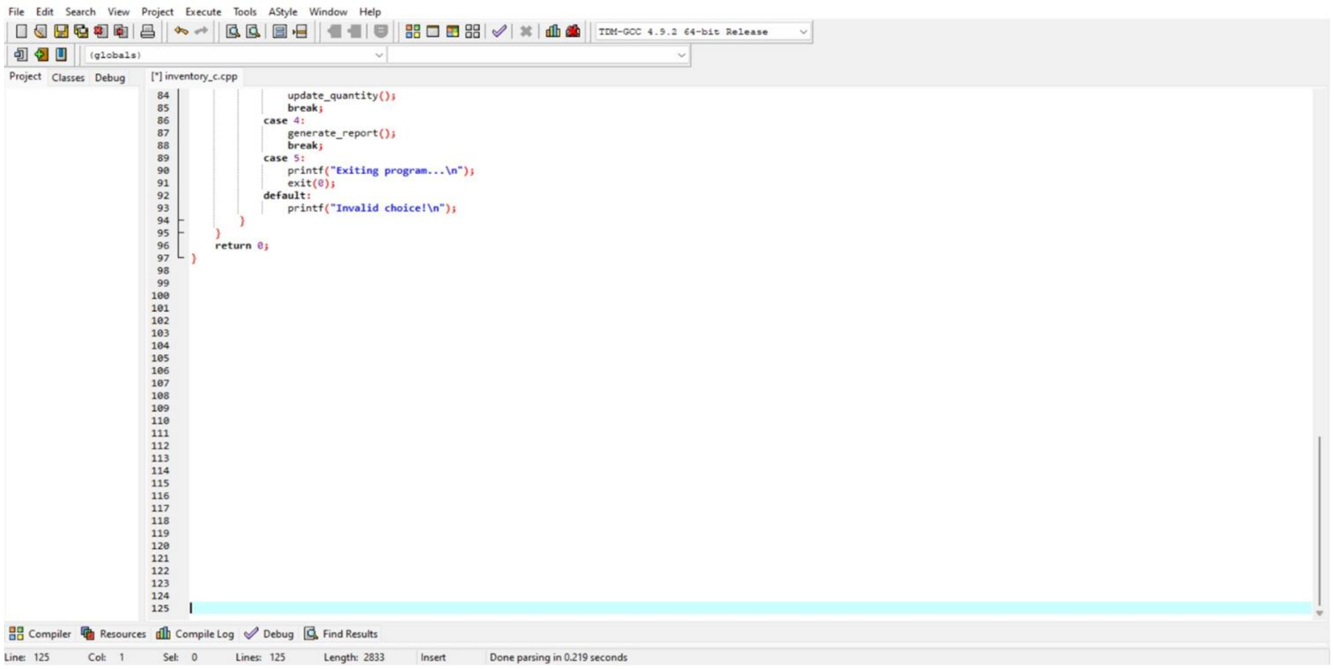
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_PRODUCTS 100

typedef struct Product {
    char name[50];
    int code;
    int quantity;
    float price;
} Product;

Product products[MAX_PRODUCTS];
int num_products = 0;

void add_product() {
    Product p;
    printf("Enter product name: ");
    scanf("%s", p.name);
    printf("Enter product code: ");
    scanf("%d", &p.code);
    printf("Enter product quantity: ");
    scanf("%d", &p.quantity);
    printf("Enter product price: ");
    scanf("%f", &p.price);
    products[num_products++] = p;
    printf("Product added successfully!\n");
}

void display_products() {
    printf("Products in inventory:\n");
    for (int i = 0; i < num_products; i++) {
        printf("%d. Name: %s | Code: %d | Quantity: %d | Price: %.2f\n", i+1, products[i].name, products[i].code, products[i].quantity, products[i].price);
    }
}

void update_quantity() {
    int code, quantity;
    printf("Enter product code: ");
    scanf("%d", &code);
    printf("Enter new quantity: ");
```

Line: 1 Col: 1 Sel: 0 Lines: 97 Length: 2777 Insert Done parsing in 0.219 seconds



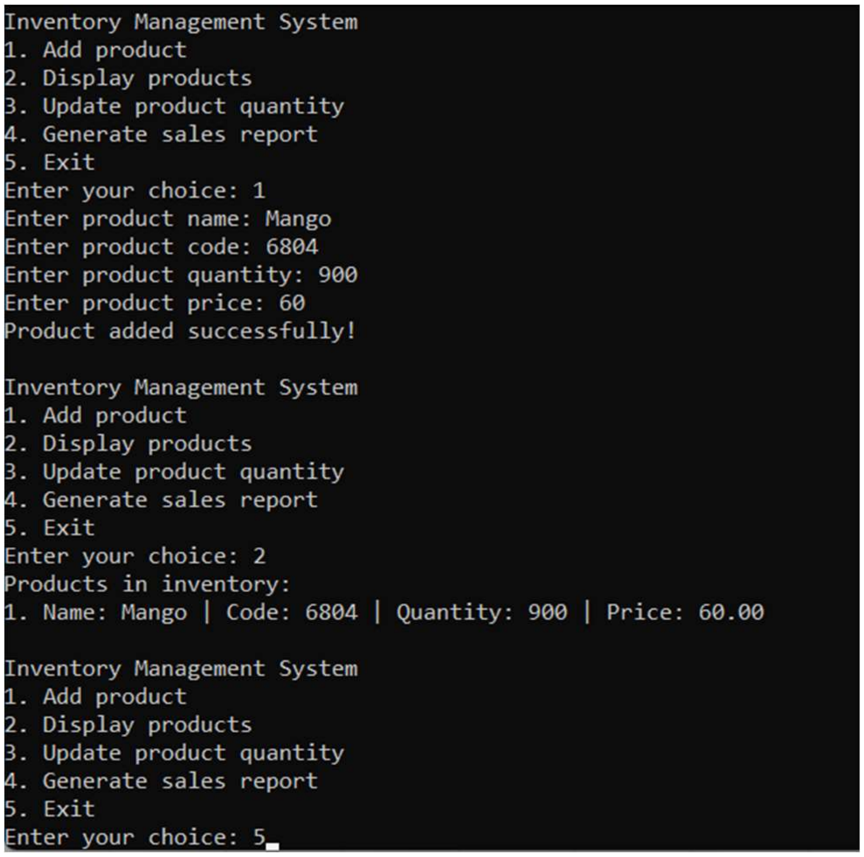
```c
    scanf("%d", &quantity);
    for (int i = 0; i < num_products; i++) {
        if (products[i].code == code) {
            products[i].quantity = quantity;
            printf("Quantity updated successfully!\n");
            return;
        }
    }
    printf("Product not found!\n");
}

void generate_report() {
    float total_sales = 0;
    printf("Sales report:\n");
    for (int i = 0; i < num_products; i++) {
        float sales = (products[i].quantity * products[i].price);
        total_sales += sales;
        printf("%d. Name: %s | Code: %d | Quantity: %d | Price: %.2f | Sales: %.2f\n", i+1, products[i].name, products[i].code, products[i].quantity, products[i].price, sales);
    }
    printf("Total sales: %.2f\n", total_sales);
}

int main() {
    int choice;
    while (1) {
        printf("\nInventory Management System\n");
        printf("1. Add product\n");
        printf("2. Display products\n");
        printf("3. Update product quantity\n");
        printf("4. Generate sales report\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                add_product();
                break;
            case 2:
                display_products();
                break;
            case 3:
                update_quantity();
```

Line: 1 Col: 1 Sel: 0 Lines: 97 Length: 2777 Insert Done parsing in 0.219 seconds



```
                update_quantity();
                break;
            case 4:
                generate_report();
                break;
            case 5:
                printf("Exiting program...\n");
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

Output :



```
Inventory Management System
1. Add product
2. Display products
3. Update product quantity
4. Generate sales report
5. Exit
Enter your choice: 1
Enter product name: Mango
Enter product code: 6804
Enter product quantity: 900
Enter product price: 60
Product added successfully!

Inventory Management System
1. Add product
2. Display products
3. Update product quantity
4. Generate sales report
5. Exit
Enter your choice: 2
Products in inventory:
1. Name: Mango | Code: 6804 | Quantity: 900 | Price: 60.00

Inventory Management System
1. Add product
2. Display products
3. Update product quantity
4. Generate sales report
5. Exit
Enter your choice: 5
```

Variable Descriptions:

`Product`: Structure representing a product, containing the following fields:

`name`: Name of the product (string).

`code`: Code of the product (integer).

`quantity`: Quantity of the product (integer).

`price`: Price of the product (float).

`products`: Array of `Product` structures to store the products in inventory.

`num_products`: Number of products currently stored in the inventory.

`choice`: User's choice of operation.

File Descriptions/Function Description:

This is a simple inventory management system implemented in C. It allows users to perform the following actions:

1. Add a product: Prompts the user to enter the name, code, quantity, and price of a product and adds it to the inventory.
2. Display products: Prints a list of all products in the inventory, including their name, code, quantity, and price.
3. Update product quantity: Prompts the user to enter a product code and a new quantity, and updates the quantity of the corresponding product in the inventory.
4. Generate sales report: Calculates and prints a sales report, including the total sales for each product (quantity * price) and the overall total sales.
5. Exit: Terminates the program.

The program uses an array of structures (`Product`) to store the products, and a variable (`num_products`) to keep track of the number of products in the inventory. The user is presented with a menu of options, and their choice is processed in a loop until they choose to exit the program.

The program uses a structure called "Product" to represent each product, with fields for name, code, quantity, and price. An array called "products" holds the inventory of products, and the variable "num_products" keeps track of the number of products currently in the inventory.

The main function presents a menu to the user and allows them to choose an action by entering a number. Depending on the choice, the corresponding function is called to perform the desired operation.

The program continues to display the menu until the user chooses to exit.

Here's an outline of a C program that can manage inventory, track product sales, manage stock, and

1. Define necessary data structures:

Product structure: includes attributes such as name, ID, price, quantity, etc.

Inventory structure: contains an array of Product structures and the total number of products.

2. Implement functions for managing inventory:

Initialize inventory: sets up an empty inventory.

Add product: allows adding a new product to the inventory.

Remove product: allows removing a product from the inventory.

Update product quantity: modifies the quantity of a specific product in the inventory.

3. Implement functions for tracking sales:

Record sale: records the sale of a product, reducing its quantity in the inventory.

4. Implement functions for generating reports:

Generate stock report: displays the current stock status, including product names, IDs, and quantities.

Generate sales report: provides information on product sales, such as total sales, revenue, and quantities sold.

5. Implement the main program loop:

Display a menu of options to the user.

Accept user input and call the corresponding functions based on the chosen option.

Continue the loop until the user chooses to exit the program.