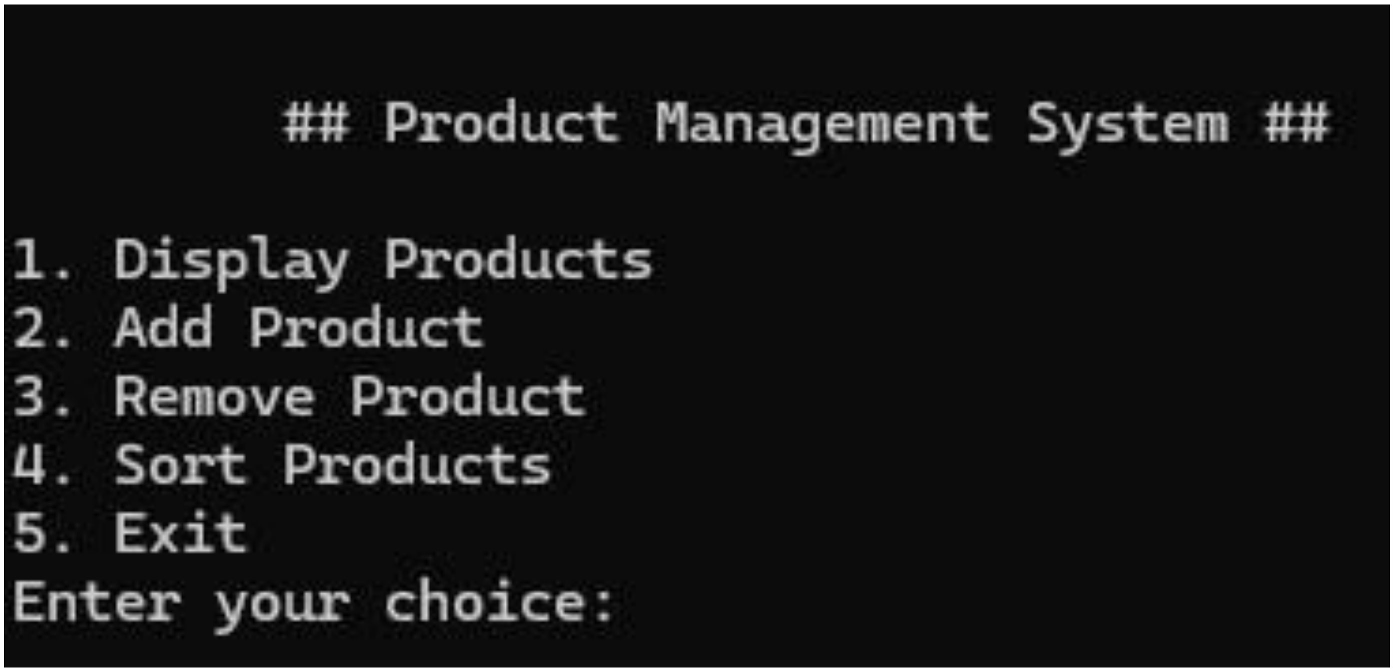
❖ **Product Management System using C++**

INTRODUCTION:

The provided C++ code implements a simple Product Management System that allows users to manage a list of products. Users can add, remove, display, and sort products based on their prices. This system is designed to demonstrate fundamental programming concepts such as structures, arrays, functions, and control flow in C++.



```
        ## Product Management System ##

1. Display Products
2. Add Product
3. Remove Product
4. Sort Products
5. Exit
Enter your choice:
```

KEY CONCEPTS

1. **STRUCTURES:** The Product structure is defined to hold product information, including the product's name and price.

2. **DYNAMIC ARRAY MANAGEMENT:**

   The program uses a static array to store products, with a variable n to track the number of products currently in the list.

3. **FUNCTIONS:**

   Several functions are defined to handle different operations: adding a product, removing a product, displaying products, and sorting products.

4. **CONTROL FLOW:**

   The program utilizes a loop and a switch statement to allow user interaction and handle different commands.

CODE STRUCTURE

The code is structured into 3 key parts:

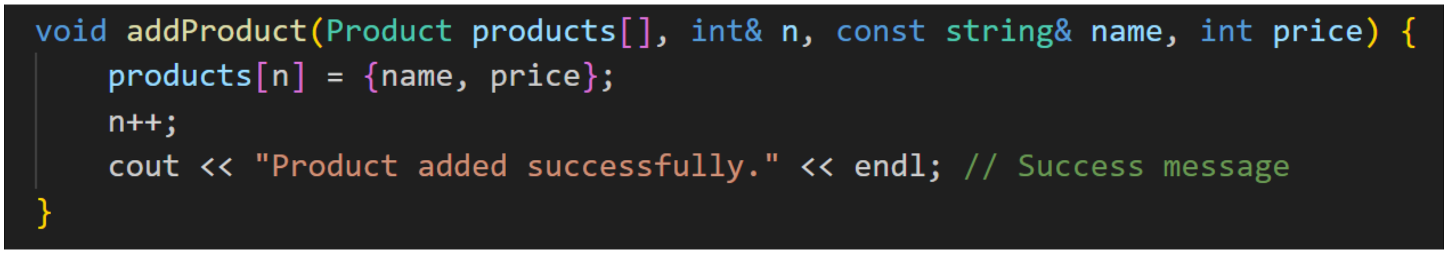
- **PRODUCT STRUCTURE:** Defines the attributes of a product.

- **FUNCTION DEFINITIONS:** Each function performs a specific task related to product management.

- **MAIN FUNCTION:** The entry point of the program, which handles user input and orchestrates the various functionalities.

MECHANISM

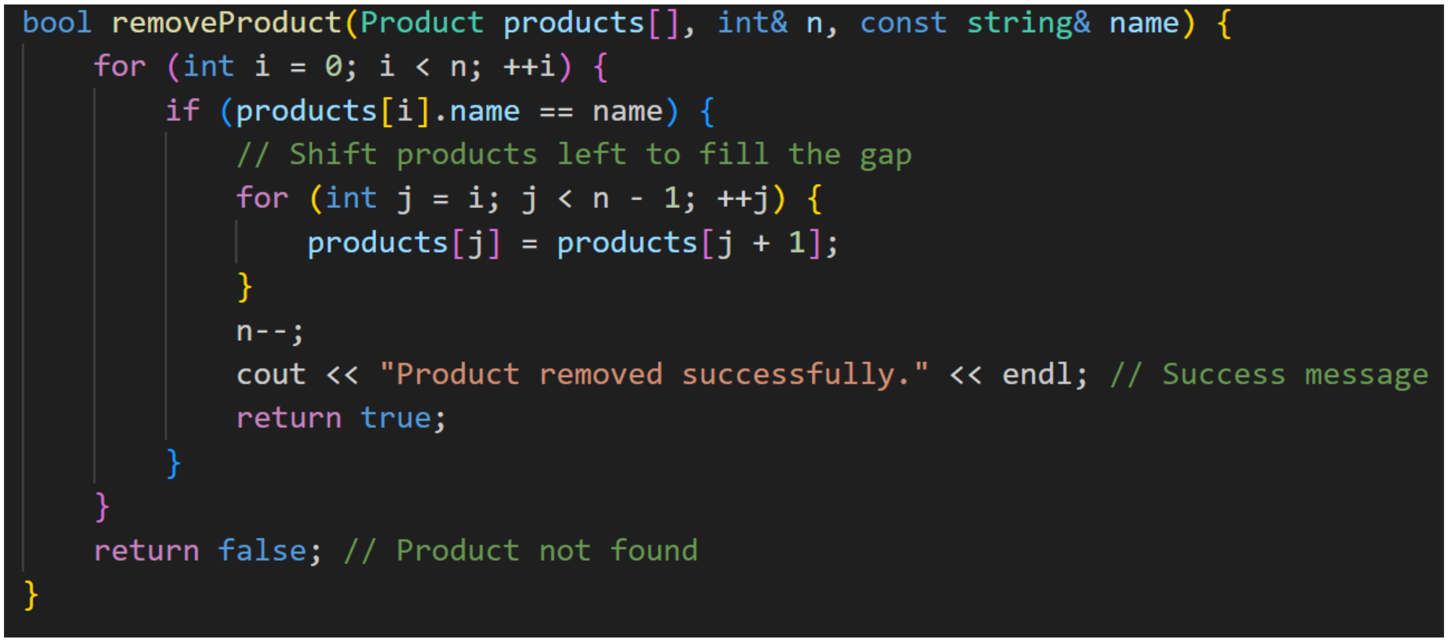
- **ADDING A PRODUCT**



```cpp
void addProduct(Product products[], int& n, const string& name, int price) {
    products[n] = {name, price};
    n++;
    cout << "Product added successfully." << endl; // Success message
}
```

We have used an array of Product structures, a reference to the number of products, the product name, and its price. It adds a new product to the array and increments the count of products.

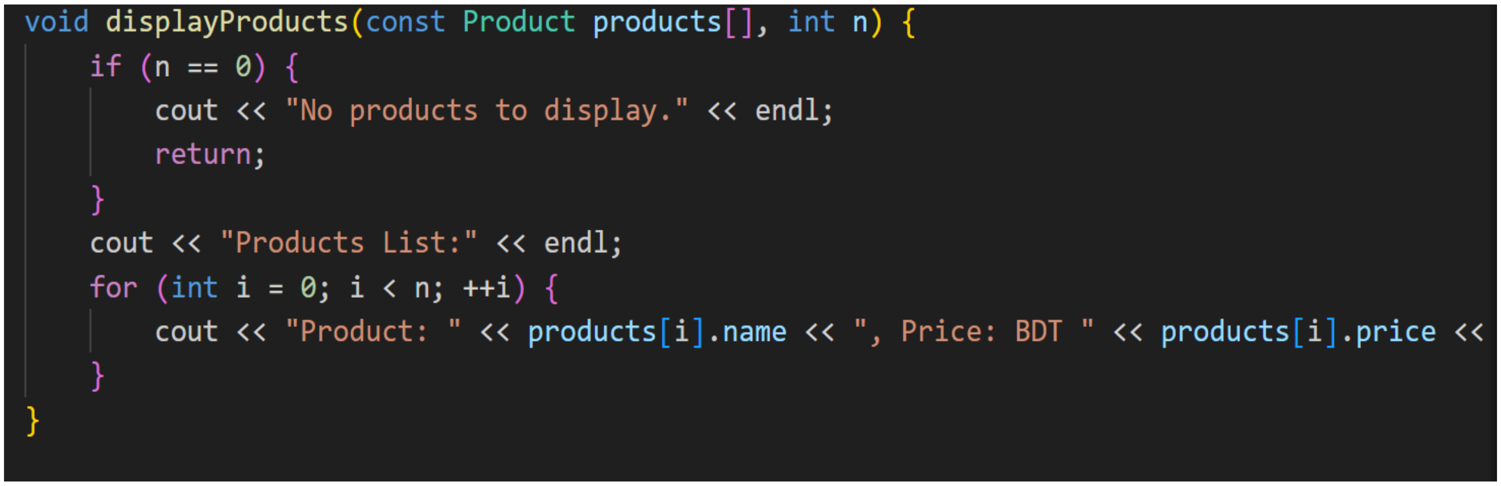
- **REMOVING PRODUCT**



```cpp
bool removeProduct(Product products[], int& n, const string& name) {
    for (int i = 0; i < n; ++i) {
        if (products[i].name == name) {
            // Shift products left to fill the gap
            for (int j = i; j < n - 1; ++j) {
                products[j] = products[j + 1];
            }
            n--;
            cout << "Product removed successfully." << endl; // Success message
            return true;
        }
    }
    return false; // Product not found
}
```

Function that searches for a product by name. If found, it shifts the remaining products to fill the gap and decrements the product count.

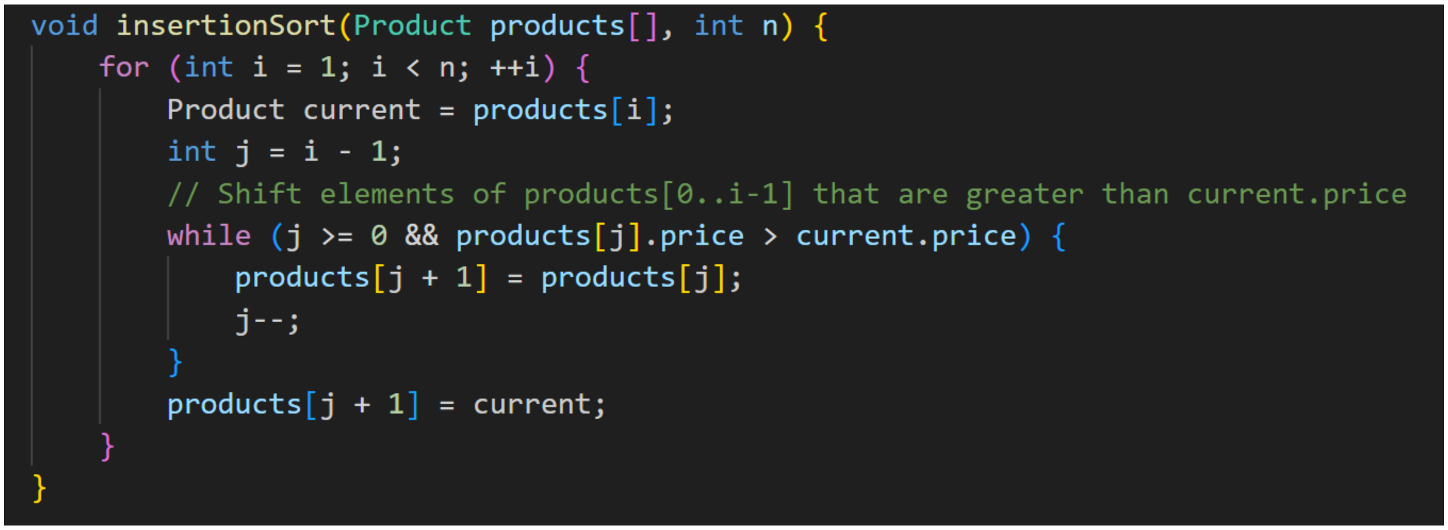
- **DISPLAYING PRODUCTS**



```
void displayProducts(const Product products[], int n) {
    if (n == 0) {
        cout << "No products to display." << endl;
        return;
    }
    cout << "Products List:" << endl;
    for (int i = 0; i < n; ++i) {
        cout << "Product: " << products[i].name << ", Price: BDT " << products[i].price <<
    }
}
```

This function displays all products in the list. If there are no products, it informs the user accordingly.

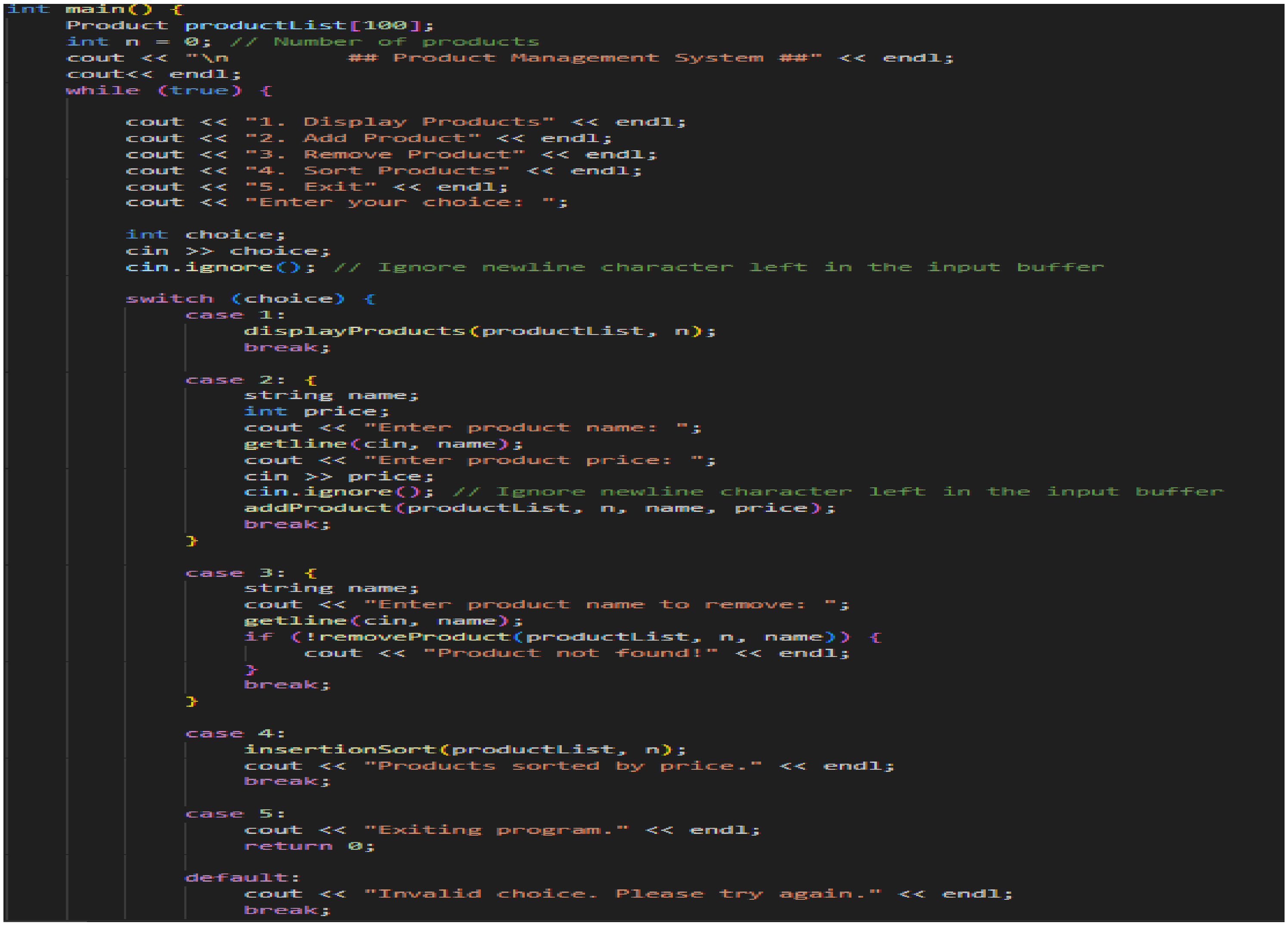
- **SORTING PRODUCTS**



```
void insertionSort(Product products[], int n) {
    for (int i = 1; i < n; ++i) {
        Product current = products[i];
        int j = i - 1;
        // Shift elements of products[0..i-1] that are greater than current.price
        while (j >= 0 && products[j].price > current.price) {
            products[j + 1] = products[j];
            j--;
        }
        products[j + 1] = current;
    }
}
```

This function sorts the products by price using the insertion sort algorithm. It iterates through the list and places each product in its correct position based on price.

- **THE MAIN FUNCTION BEHIND SMOOTH FUNCTIONALITY**



```
int main() {
    Product productList[100];
    int n = 0; // Number of products
    cout << "\n          ## Product Management System ##" << endl;
    cout<< endl;
    while (true) {

        cout << "1. Display Products" << endl;
        cout << "2. Add Product" << endl;
        cout << "3. Remove Product" << endl;
        cout << "4. Sort Products" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";

        int choice;
        cin >> choice;
        cin.ignore(); // Ignore newline character left in the input buffer

        switch (choice) {
            case 1:
                displayProducts(productList, n);
                break;

            case 2: {
                string name;
                int price;
                cout << "Enter product name: ";
                getline(cin, name);
                cout << "Enter product price: ";
                cin >> price;
                cin.ignore(); // Ignore newline character left in the input buffer
                addProduct(productList, n, name, price);
                break;
            }

            case 3: {
                string name;
                cout << "Enter product name to remove: ";
                getline(cin, name);
                if (!removeProduct(productList, n, name)) {
                    cout << "Product not found!" << endl;
                }
                break;
            }

            case 4:
                insertionSort(productList, n);
                cout << "Products sorted by price." << endl;
                break;

            case 5:
                cout << "Exiting program." << endl;
                return 0;

            default:
                cout << "Invalid choice. Please try again." << endl;
                break;
```

Arrays, Loops, Switch Statements, Functions, cin & getline for user input has been used in the main function.

CONCLUSION

The Product Management System implemented in C++ provides a clear and concise way to manage a list of products. Through the use of structures, functions, and control flow, the code demonstrates essential programming concepts that are foundational for developing more complex applications. This system can be further enhanced by incorporating features such as file storage, user authentication, or a graphical user interface, making it a versatile starting point for aspiring developers.

THANK YOU

FOR YOUR VALUABLE TIME