**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

1. **Understand Asymptotic Notation:**

**Explain Big O notation and how it helps in analyzing algorithms.**

**Big O Notation** describes how the performance (time or space) of an algorithm grows as the input size increases.

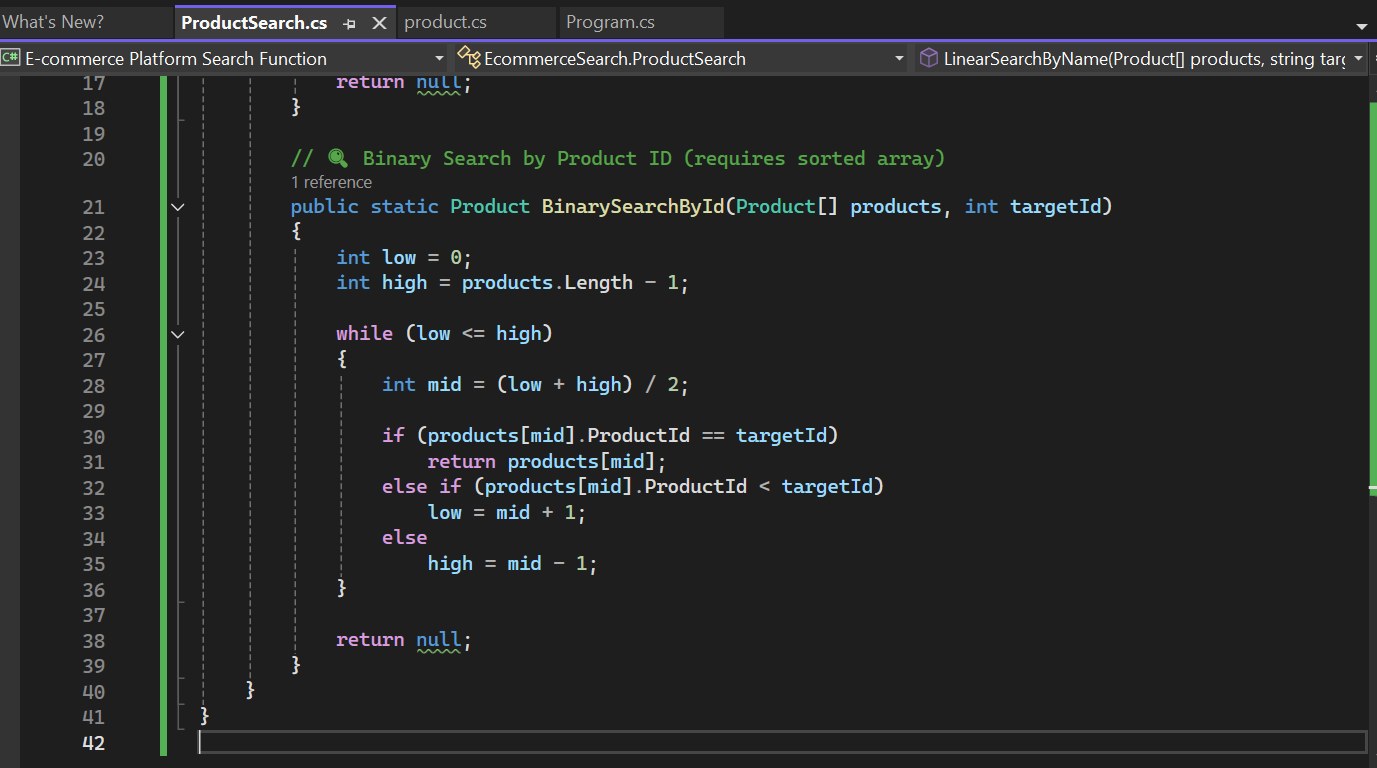
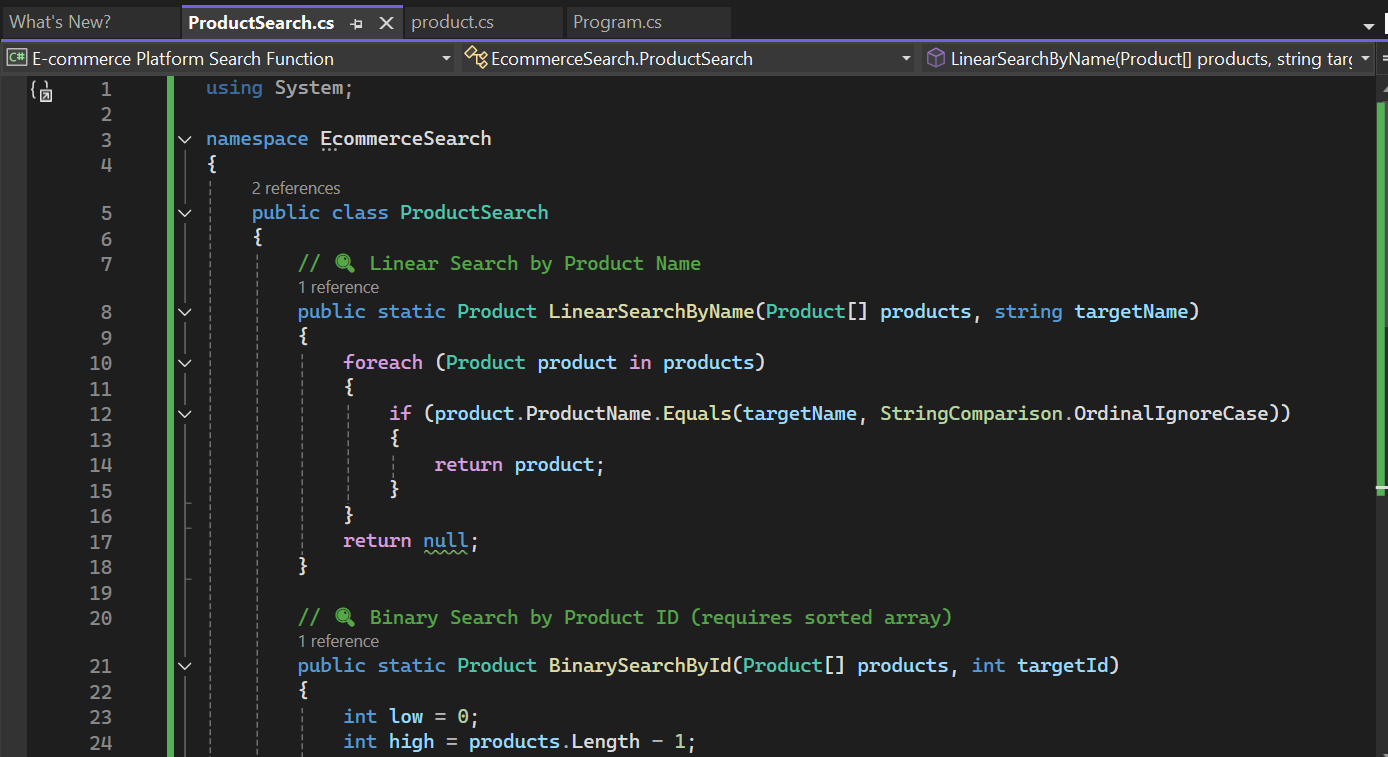
| **Notation** | **Meaning** | **Example Algorithm** |
| --- | --- | --- |
| O(1) | Constant Time | Accessing an array item by index |
| O(n) | Linear Time | Linear Search |
| O(log n) | Logarithmic Time | Binary Search |
| O(n²) | Quadratic Time | Bubble Sort |

**2.Describe the best, average, and worst-case scenarios for search operations.**

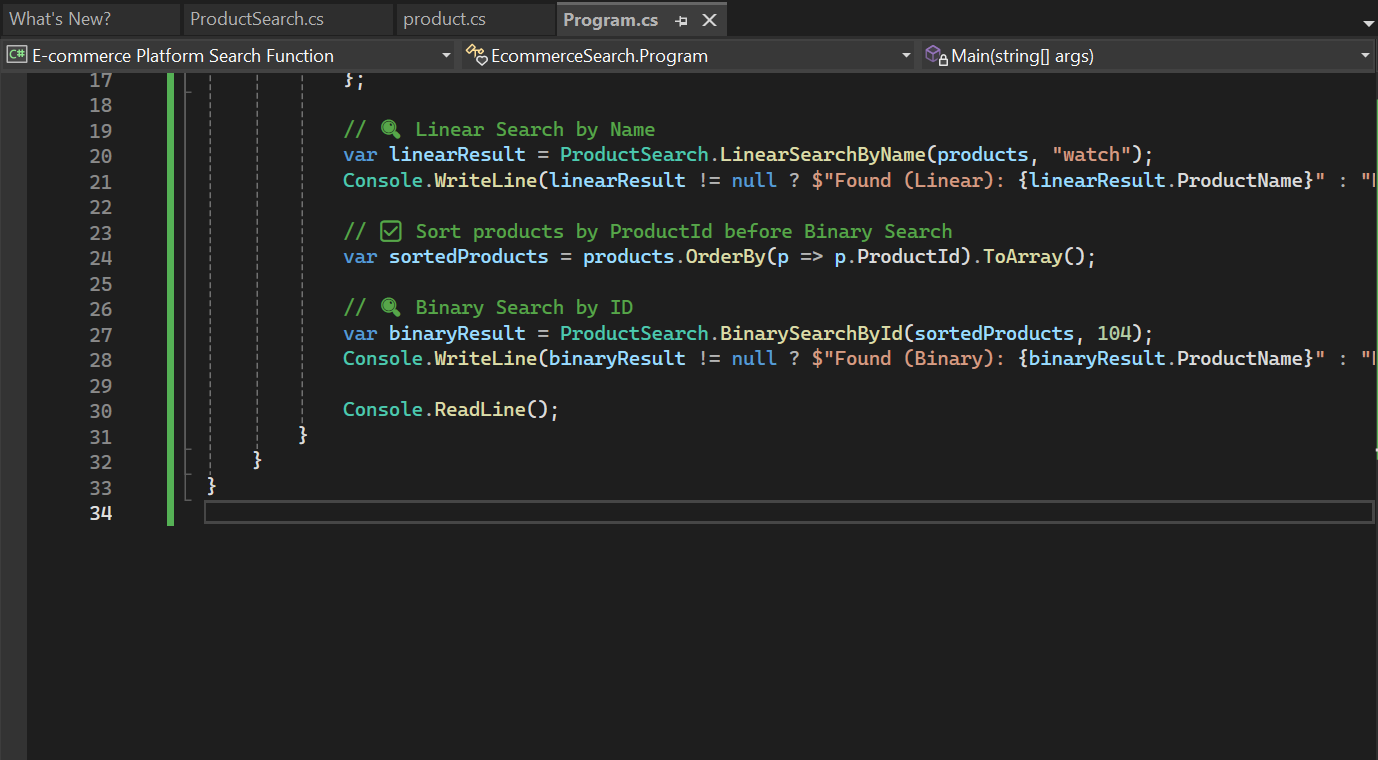
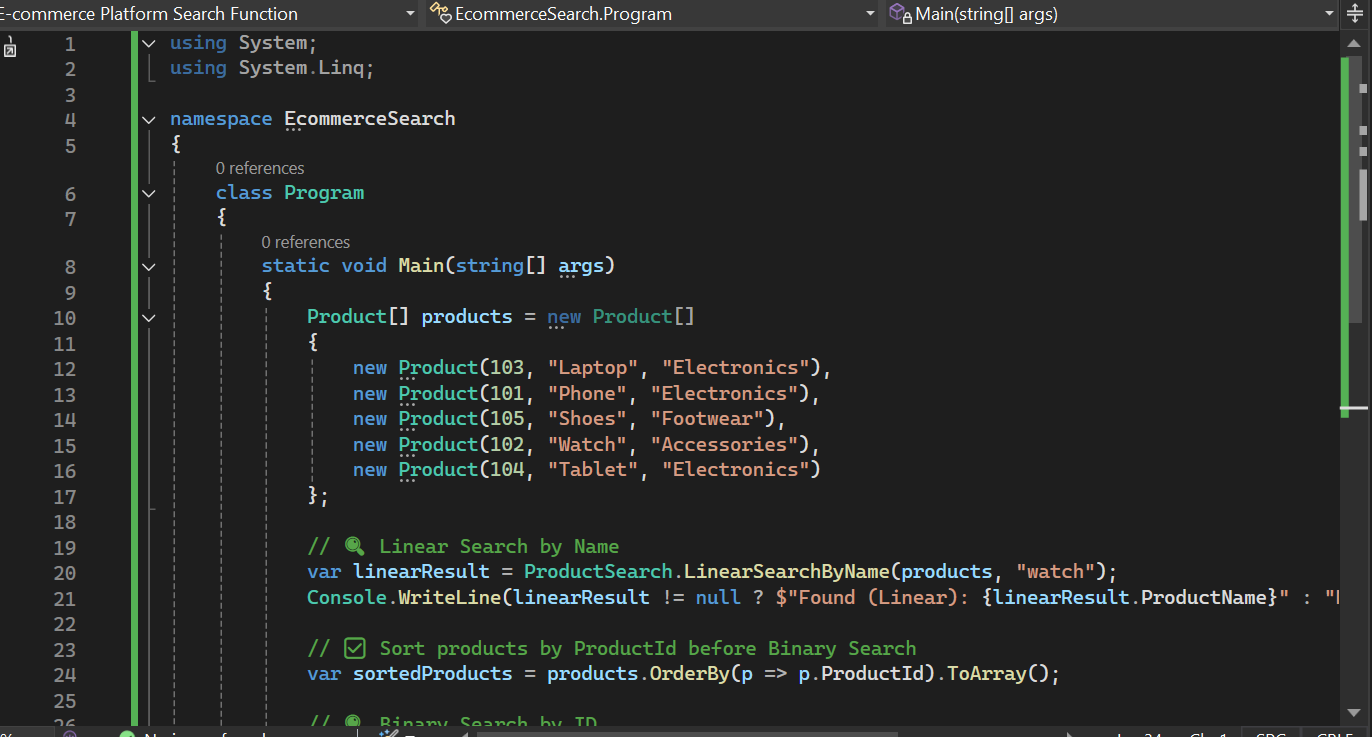
In search operations, the best, average, and worst-case scenarios describe how efficiently an algorithm performs under different input conditions. The **best-case scenario** occurs when the target element is found immediately, such as at the beginning of a list, resulting in minimal time complexity — often O(1) for linear search. The **average-case scenario** represents the expected time an algorithm takes across many random inputs; for linear search, it typically takes O(n/2), which simplifies to O(n), assuming the element is equally likely to appear anywhere. The **worst-case scenario** arises when the target is at the end of the list or not present at all, requiring the algorithm to scan the entire data set — again resulting in O(n) time for linear search. In contrast, optimized search algorithms like binary search (used on sorted data) offer better performance, with best-case O(1), average-case O(log n), and worst-case O(log n), showcasing the importance of algorithm choice based on data structure and context.

**Implementation:**

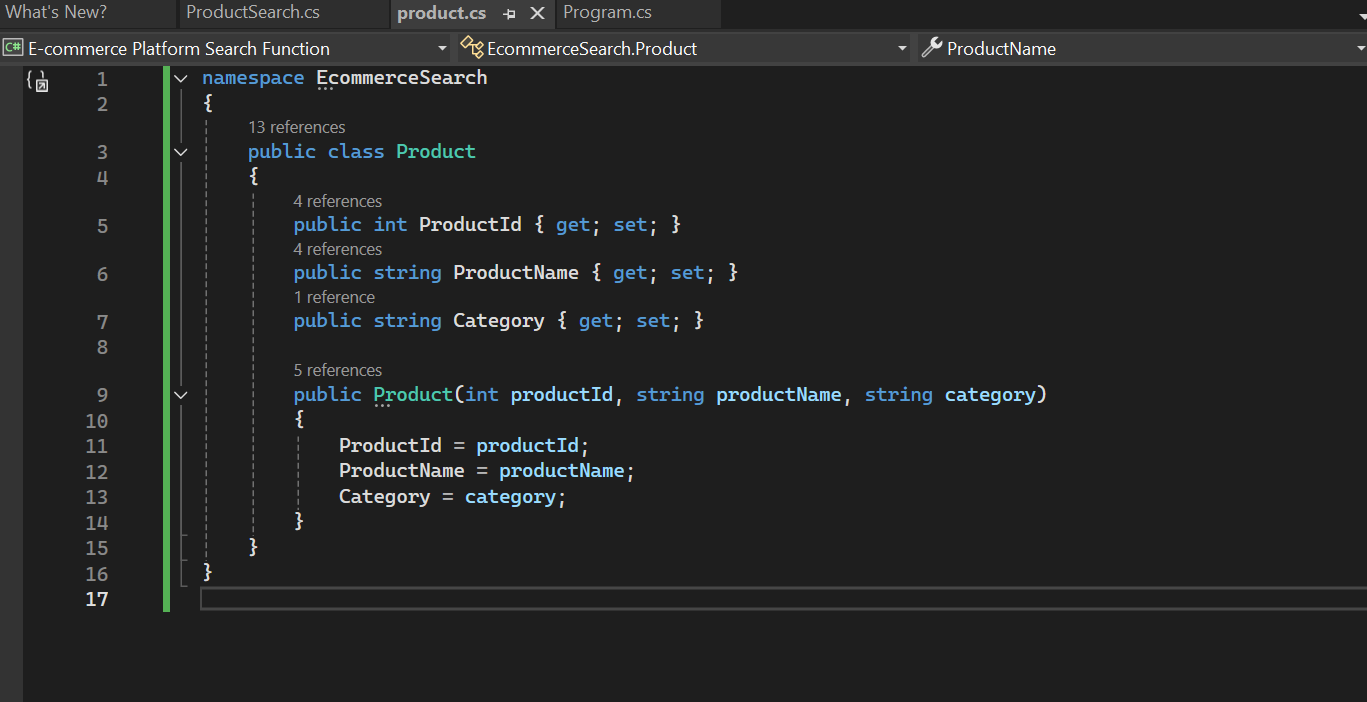
Producttsearch.cs

****

Program.cs

****

Product.cs

****

**Analysis:**

**Compare the time complexity of linear and binary search algorithms.**

Linear search and binary search are two ways to find an element in a list. Linear search checks each item one by one, so it takes more time, especially if the list is long. Its best case is O(1), but the average and worst cases are O(n). Binary search is faster, with a time complexity of O(log n) for average and worst cases, but it only works on sorted lists. It keeps dividing the list in half to find the target. So, binary search is much quicker than linear search for large, sorted data.

**Discuss which algorithm is more suitable for your platform and why.**

For an e-commerce platform, **binary search** is more suitable because it offers much faster performance when dealing with large product datasets. Since e-commerce platforms often organize and sort products by name, price, or category, using binary search on sorted data allows quicker results, improving user experience. Unlike linear search, which checks each product one by one, binary search reduces the number of comparisons significantly, making it ideal for search operations where speed and efficiency matter. However, maintaining sorted data is essential for binary search to work correctly.