**Superset ID :**  6365409

**Data structures and Algorithms**

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

1. Big O notation is used to describe the performance (time or space complexity) of an algorithm as the input size increases.

Why it's useful:

* It helps to understand how scalable an algorithm is.
* Helps compare performance of different algorithms.
* Guides decisions for optimization.

Search Case Scenarios:

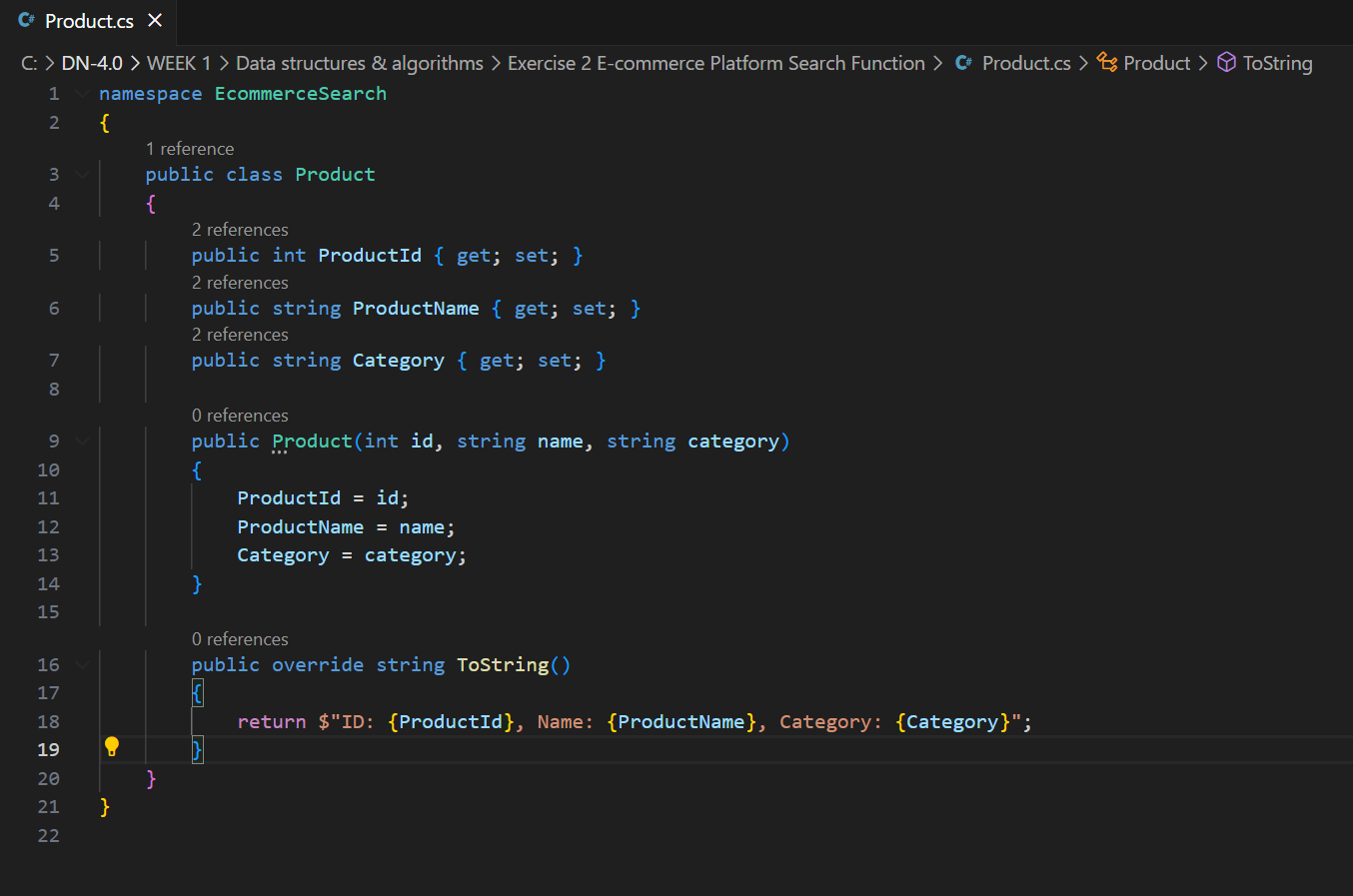
Linear Search:

* Best Case: O(1) when the target is the first element.
* Average Case: O(n)
* Worst Case: O(n)

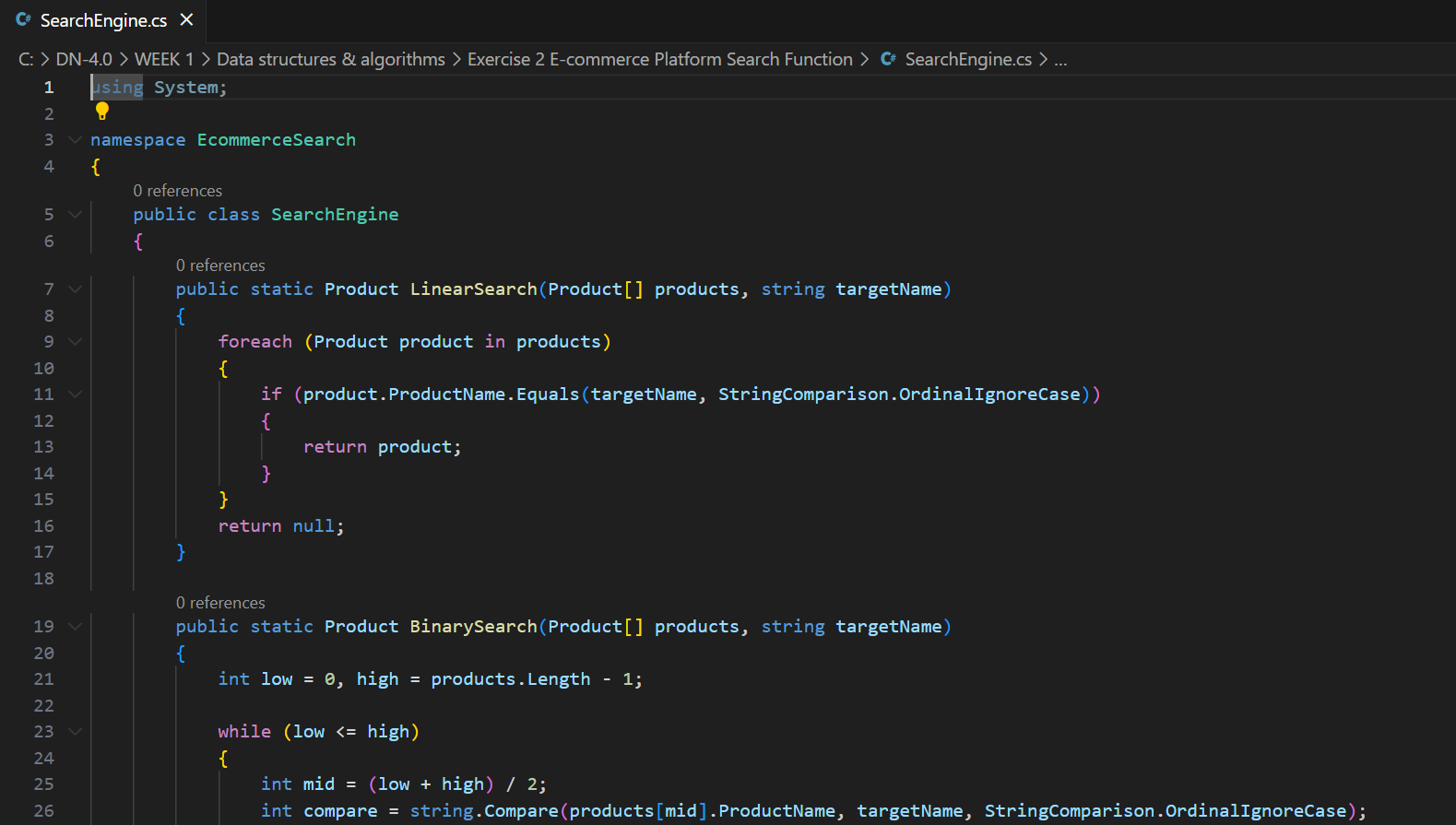
Binary Search:

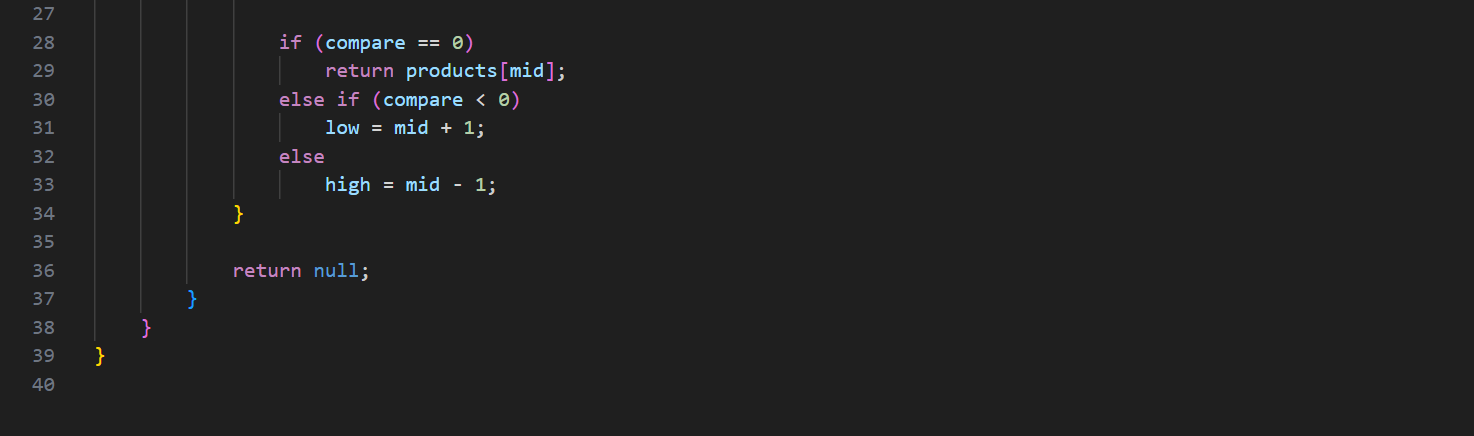
* Best Case: O(1) if the target is the middle element.
* Average Case: O(log n)
* Worst Case: O(log n)

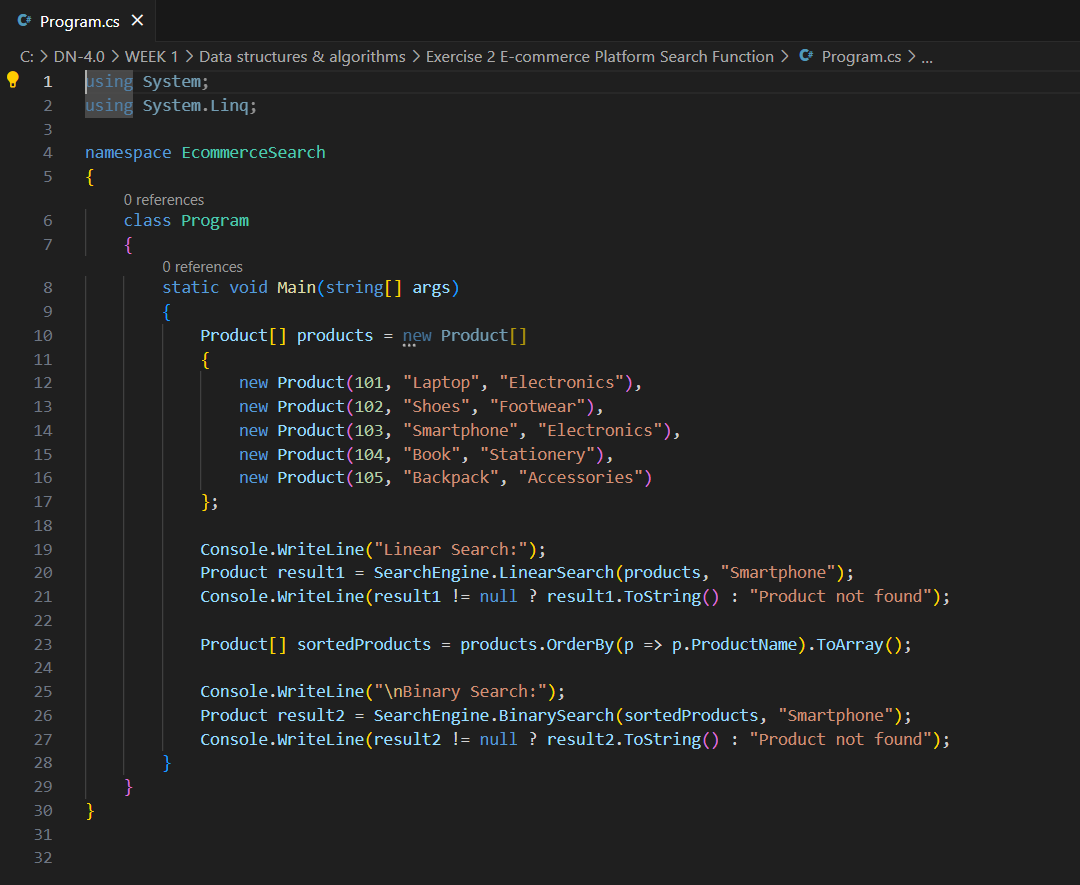
1. Product.cs



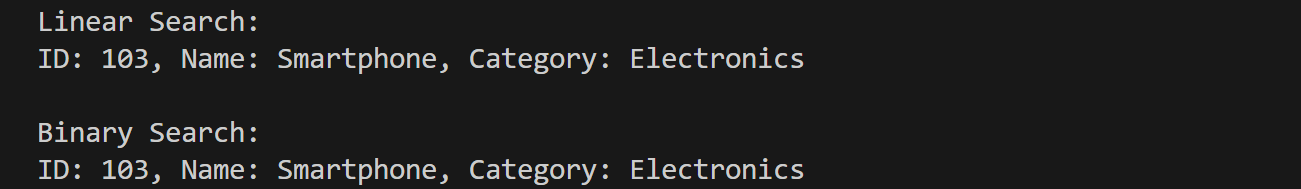
SearchEngine.cs





Program.cs

Output :



**Time Complexity Analysis**

Linear Search has time complexity O(n). It works for small or unsorted data.

Binary Search has time complexity O(log n). It is faster and works when the data is sorted.

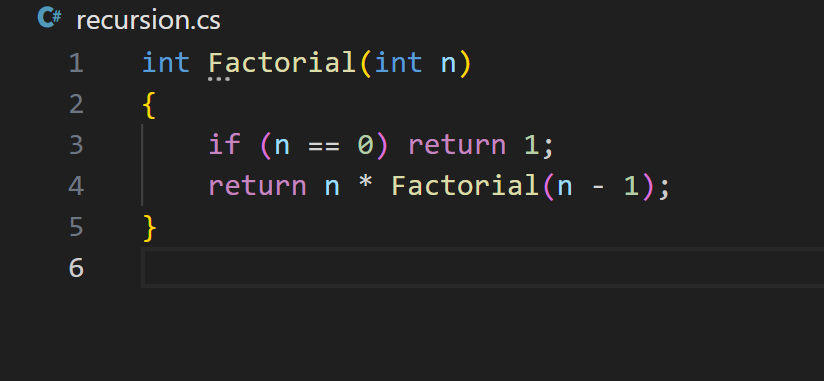
**Recommendation**:

For an e-commerce platform where fast searching is needed and the data can be sorted or indexed, binary search is more suitable. In large-scale systems, further optimizations like indexing or database-level search may be applied.

**Exercise 7: Financial Forecasting**

1. Recursion is when a method calls itself to solve smaller instances of the same problem. It helps simplify problems like factorial, Fibonacci, and tree traversals.

Example :



In financial forecasting, recursion can model future values based on past data, like compound interest.

1. **Method Setup (Recursive Forecast Formula)**

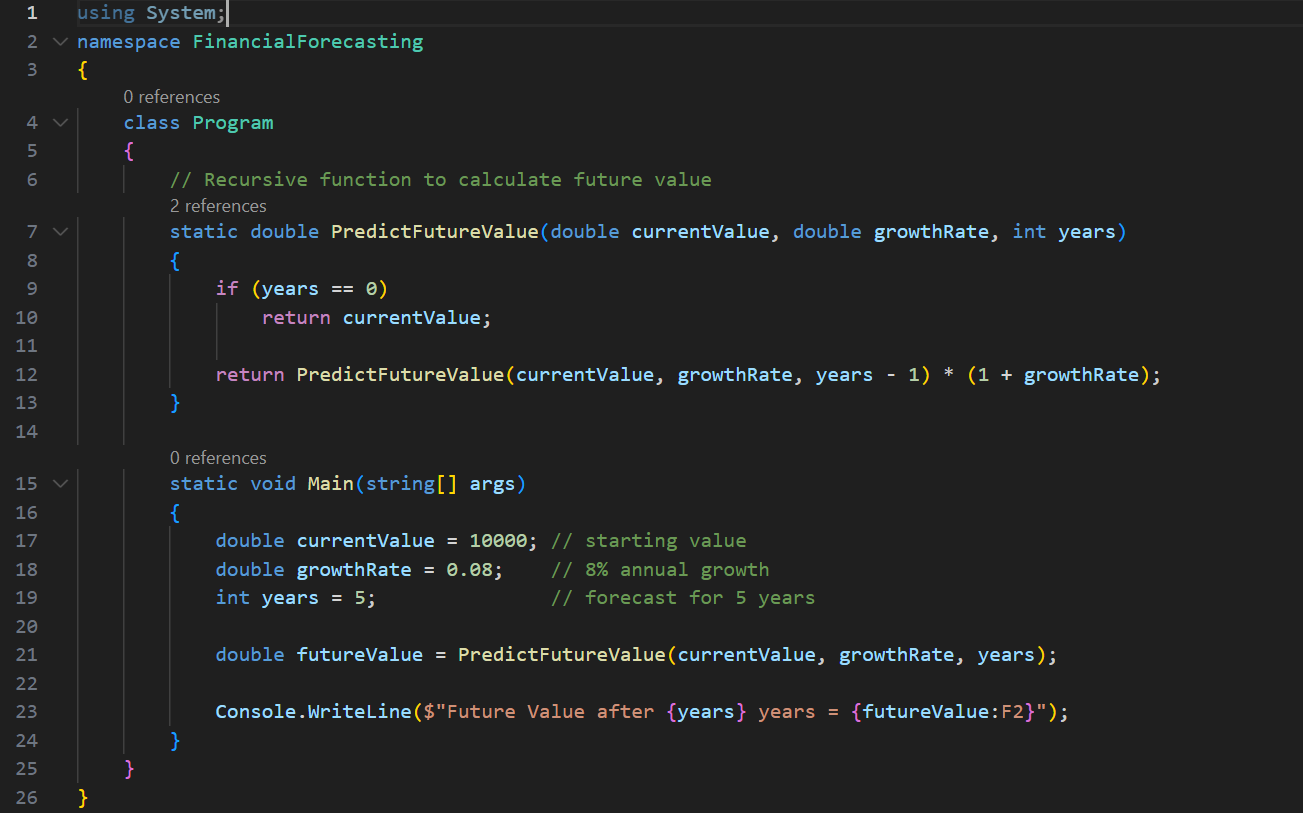
**Formula:** FutureValue(years) = CurrentValue \* (1 + growthRate)^years

We can write this as:

FV(n) = FV(n-1) \* (1 + growthRate)

Base case: FV(0) = currentValue

### Implementation



### 4. Time Complexity & Optimization

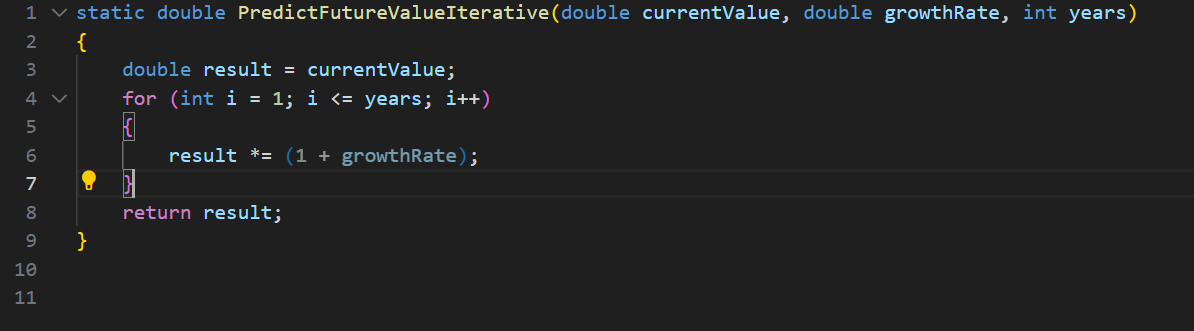
Time Complexity:

* Recursive function has O(n) time complexity where n is the number of years.
* It makes one recursive call per year.

Optimization:

* To avoid stack overflows or repeated calls, recursion can be replaced with iteration (tail recursion or loop).
* Or use **memoization** to store intermediate results.

Optional Iterative Version (More Efficient)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Thankyou\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*