**Assignment 8**

**Implement Backward Chaining algorithm**

**Theory -**

Backward chaining is a method of reasoning used in artificial intelligence, often described as a "goal-driven" or "top-down" approach. It's the counterpart to forward chaining. Instead of starting with facts to see what can be concluded, backward chaining starts with a potential conclusion (a goal) and works backward to see if it can be proven true using the available facts and rules.

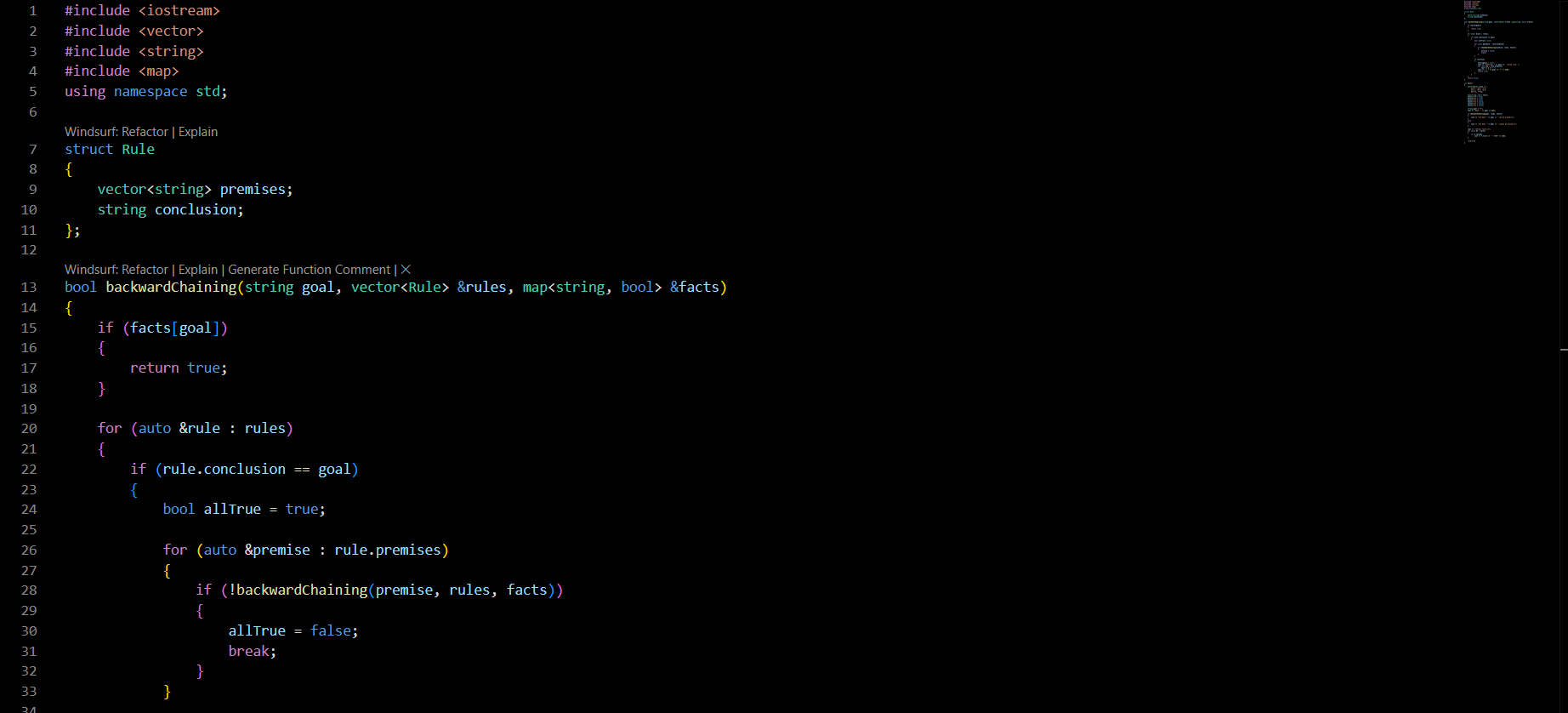
The process involves these main components:

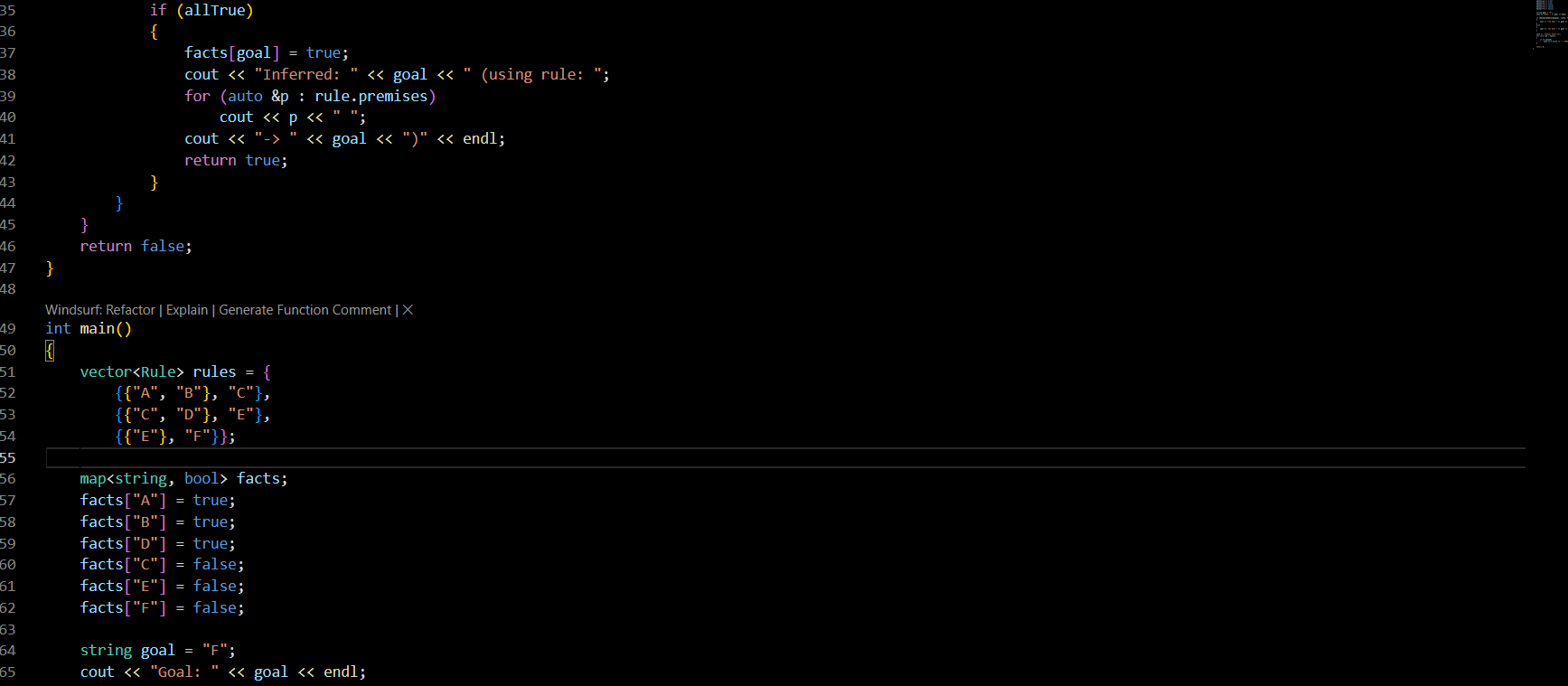
* Knowledge Base: A collection of known facts.
* Rule Base: A set of IF-THEN rules.
* Goal: The hypothesis or conclusion that the system is trying to prove.

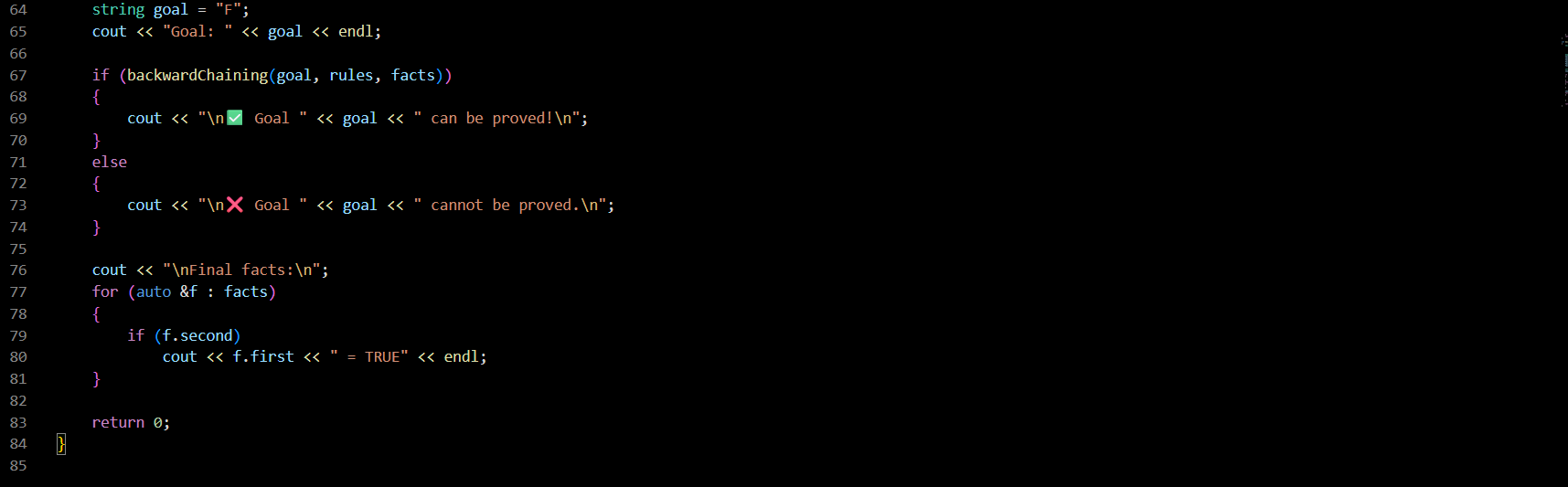
The backward chaining algorithm is inherently recursive and works as follows:

1. Start with the Goal: The system identifies the goal it needs to prove.
2. Check Facts: It first checks if the goal is already a known fact in the knowledge base. If it is, the goal is proven.
3. Find Relevant Rules: If the goal is not a known fact, the system searches the rule base for rules that have the goal as their conclusion.
4. Establish Subgoals: For each such rule, its premises become new subgoals. The system then recursively tries to prove each of these subgoals using the same backward chaining logic.
5. Prove or Fail: If all the premises (subgoals) of a rule can be proven true, then the original goal is also proven true. If the system exhausts all possible rules and cannot prove the subgoals, the original goal is considered unproven.

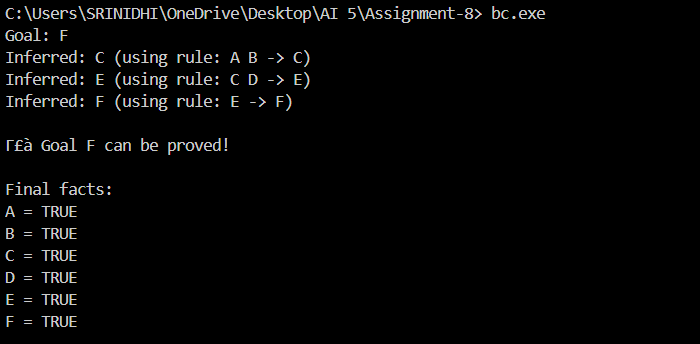
**Code –**

****

****

****

**Output –**

****

**Explanation –**

**Data Structures**

The program uses the same basic data structures as the forward chaining example:

* A Rule struct holds the premises and conclusion for each IF-THEN rule.
* A C++ map acts as the knowledge base, storing facts and their truth values.

**The backwardChaining Function**

This is the core recursive function that performs the goal-driven search. It takes the current goal it's trying to prove as an argument, along with the rule base and the fact base.

* Base Case: The function first checks if the goal is already present as a true value in the facts map. If it is, the search is successful for this branch, and the function returns true. This stops the recursion.
* Recursive Step: If the goal is not a known fact, the function iterates through all the rules to find one whose conclusion matches the current goal.
  + When it finds a matching rule, it then treats all of that rule's premises as new subgoals.
  + It makes a recursive call to itself for each premise.
  + If any of these recursive calls return false, it means that a required subgoal could not be proven. The program then gives up on the current rule and continues searching for another rule that might prove the goal.
* Inference: If the recursive calls for *all* premises of a rule return true, it means the rule's conditions are met. The program then infers that the original goal is true, updates the facts map accordingly, prints the rule that was used for the inference, and returns true.

**The main Function**

The main function sets up the problem and initiates the process.

* Initialization: It defines the rule base and the initial set of known facts.
* Set the Goal: It specifies the final conclusion it wants to verify (in this case, the goal is "F").
* Start the Engine: It makes the first call to the backwardChaining function with the initial goal.
* Report Results: Based on the boolean value returned from the function, it prints a final message indicating whether the goal could be proven. It then displays the final state of the knowledge base, which includes the initial facts plus any new facts that were inferred during the process.

**Conclusion –**  We have implemented backward chaining algorithm.