**Assignment 7**

**Implement Forward Chaining Algorithm**

**Theory -**

Forward chaining is a method of reasoning used in artificial intelligence, particularly in expert systems. It's often described as a "data-driven" or "bottom-up" approach because it starts with available data (facts) and works its way "forward" to reach a conclusion.

The process involves two main components:

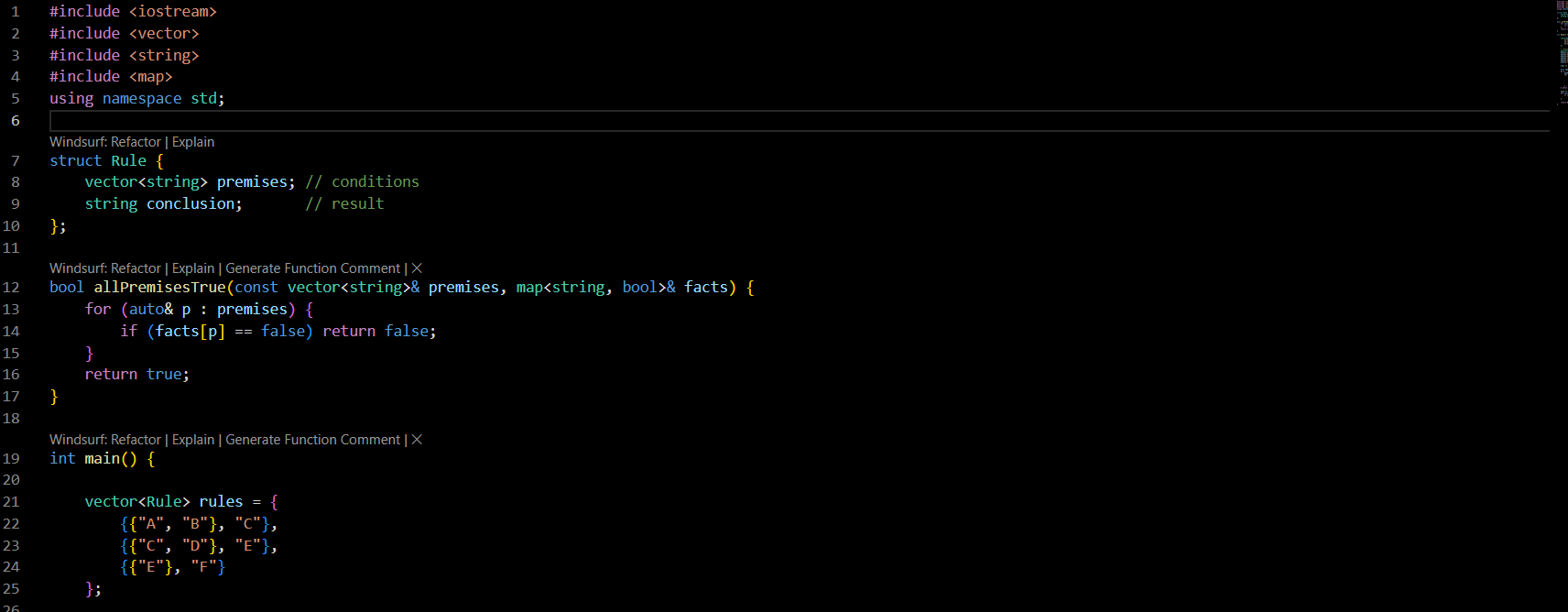
* Knowledge Base: A collection of true statements or known facts about the world.
* Rule Base: A set of IF-THEN rules (also called productions or implications). The "IF" part contains one or more conditions called premises, and the "THEN" part contains the conclusion.

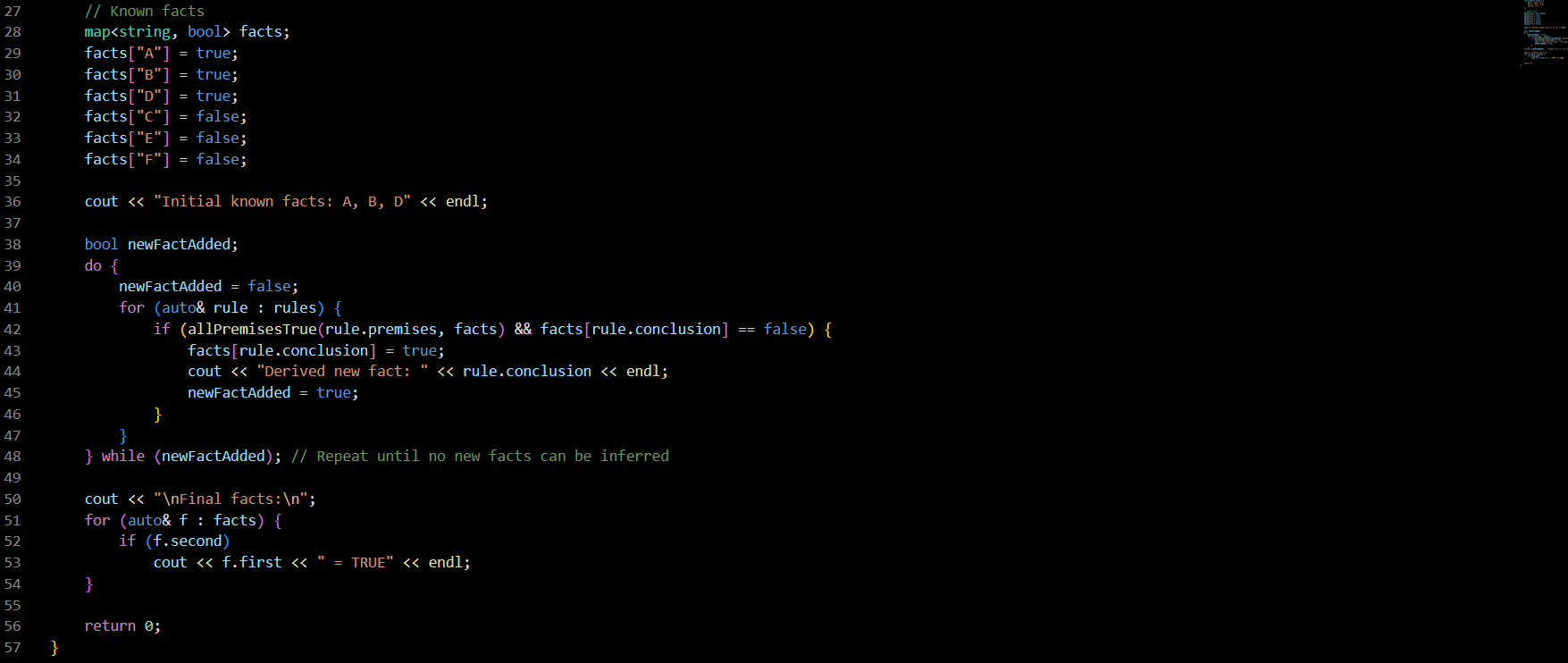
The forward chaining algorithm works in a cycle:

1. Match: The system compares the facts in the knowledge base against the premises of all the rules in the rule base.
2. Conflict Resolution: If multiple rules have their premises satisfied (i.e., they are ready to "fire"), the system must decide which one to execute first. (The provided simple code iterates in a fixed order).
3. Act: The chosen rule is fired, and its conclusion is added as a new fact to the knowledge base.
4. Repeat: This cycle repeats, using the newly added facts to potentially satisfy more rules. The process stops when no new facts can be derived.

Think of it like a detective who has a set of clues (facts) and a book of deductions (rules). The detective finds a deduction where all the required clues are present, makes the deduction, and adds it as a new clue. This new clue might then help solve another deduction, and so on.

**Code –**





**Output –**

****

**Code Explanation -**

This C++ program provides a simple implementation of a forward chaining inference engine. It starts with a few known facts and a set of rules, and then it derives every possible conclusion.

**Data Structures**

The program uses two primary data structures to represent the system:

* A Rule struct is defined to model an IF-THEN rule. It contains a list of strings for its premises and a single string for its conclusion.
* A C++ map serves as the knowledge base, storing known facts as string keys and their truth value (true or false) as the value. This allows for quick lookups to check if a fact is true.

**Helper Function**

A helper function is used to check if a rule is ready to be fired. It takes a rule's list of premises and the current knowledge base (the facts map). It iterates through the premises and returns true only if every single one is currently marked as true in the knowledge base.

**Main Logic**

The main function orchestrates the entire reasoning process.

* Initialization: The program first sets up the initial state. It defines a rule base with a few logical rules and a knowledge base with a starting set of facts. For example, it establishes that facts "A", "B", and "D" are initially true.
* The Inference Engine: The core of the program is a do-while loop that acts as the inference engine. This loop continues to run as long as new facts are being derived.
* Rule Firing Cycle: Inside the loop, the program iterates through every rule in its rule base. For each rule, it uses the helper function to check if all its premises are true. If they are, it also checks if the rule's conclusion is not already known to be true (to avoid doing redundant work).
* Deriving New Facts: If a rule's premises are met and its conclusion is new, the rule "fires." The program updates the knowledge base by setting the conclusion's value to true, prints a message indicating a new fact has been derived, and sets a flag that tells the main loop to run at least one more time.
* Termination: The loop stops when it completes a full pass through all the rules without deriving any new facts. At this point, the system is considered "stable." Finally, the program prints a list of all facts that are true in the knowledge base, which includes both the initial facts and all the newly inferred ones.

**Conclusion -**

This code effectively demonstrates the forward chaining algorithm. It shows how a system can start with a small set of data and use logical rules to expand its knowledge base, discovering new information step-by-step.

This data-driven approach is highly useful in applications where you need to react to new information as it becomes available. Common examples include diagnostic systems (e.g., "if the engine is overheating and there's smoke, then the head gasket might be blown"), monitoring applications, and planning systems that determine the consequences of a set of actions. It stands in contrast to backward chaining, a goal-driven approach that starts with a conclusion and works backward to see if it can be proven from the available facts.