Documentation:

FA:

- Has a function which reads from file a structure and it saves its values in the internal structure of Finite Automata
- Has a function which checks if an FA is either a DFA or a NDFA. It does that by checking in the map if there are duplicates between the keys of the map. The keys are formed by a pair<string, string>. If the FA has a duplicate key, then it is a NDFA, if it has no duplicate keys => FA is a DFA.
- Has a function that verifies if a sequence is accepted by the FA if FA is deterministic. It does that by parsing the states, starting from the initial state, and then goes to the next state for which it finds the current symbol from the sequence. If it reaches the empty sequence and the current state is a final state => the sequence is accepted by the FA (DFA).
- The state of FA is saved in the following variables:
    - A string for the initial state
    - A vector<string> for the final states
    - A vector<string> for the states that are not final or initial
    - A map<pair<string,string>,string> which keeps track of the transitions
    - A vector<string> which holds the alphabet
    - It has a boolean variable which states if the FA is a DFA or a NDFA.

Scanner integration:

- The integration part for the FA with the scanner is made by adding the class FA to the Scanner project.
- I create 3 FA's in the scanner class, one for identifiers (read from ids.txt file) , one for integers (read from nums.txt file) and one for doubles (read from reals.txt file).
- The checking happens in the scan function. The regex that I've previously for checking if a string obeys the specific rules, are being replaced by the specific finite automata check's function.