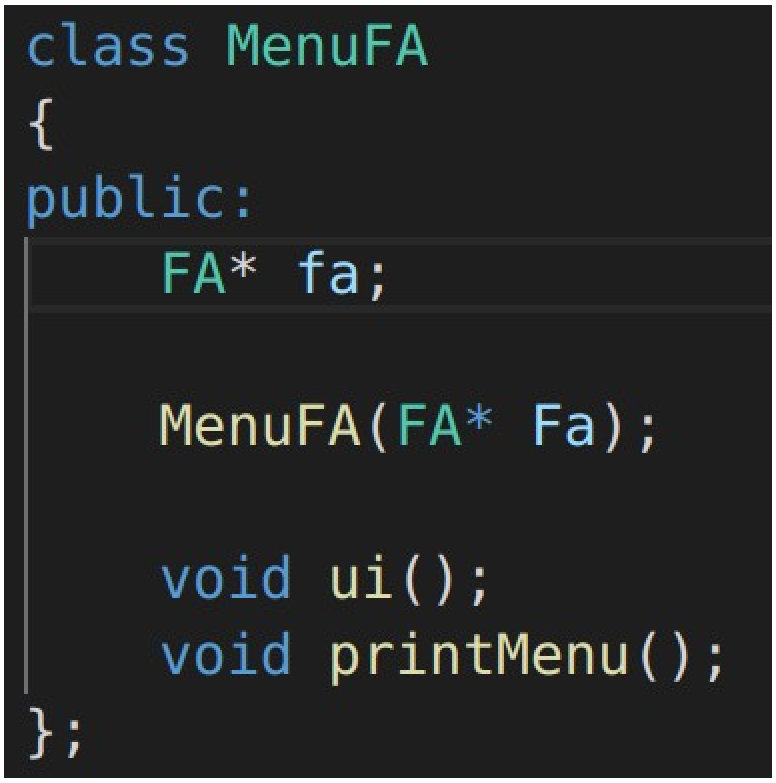
Chis Sergiu
Group: 932/1

Lab4

Github: https://github.com/SergiuChis/FLCD/tree/lab4
(the link goes directly to the 'lab4' branch)

For the implementation of this laboratory, 3 classes were written: MenuFA, Fa, and Transition.

The MenuFA class handles all the interactions with the user. It is defined as follows:



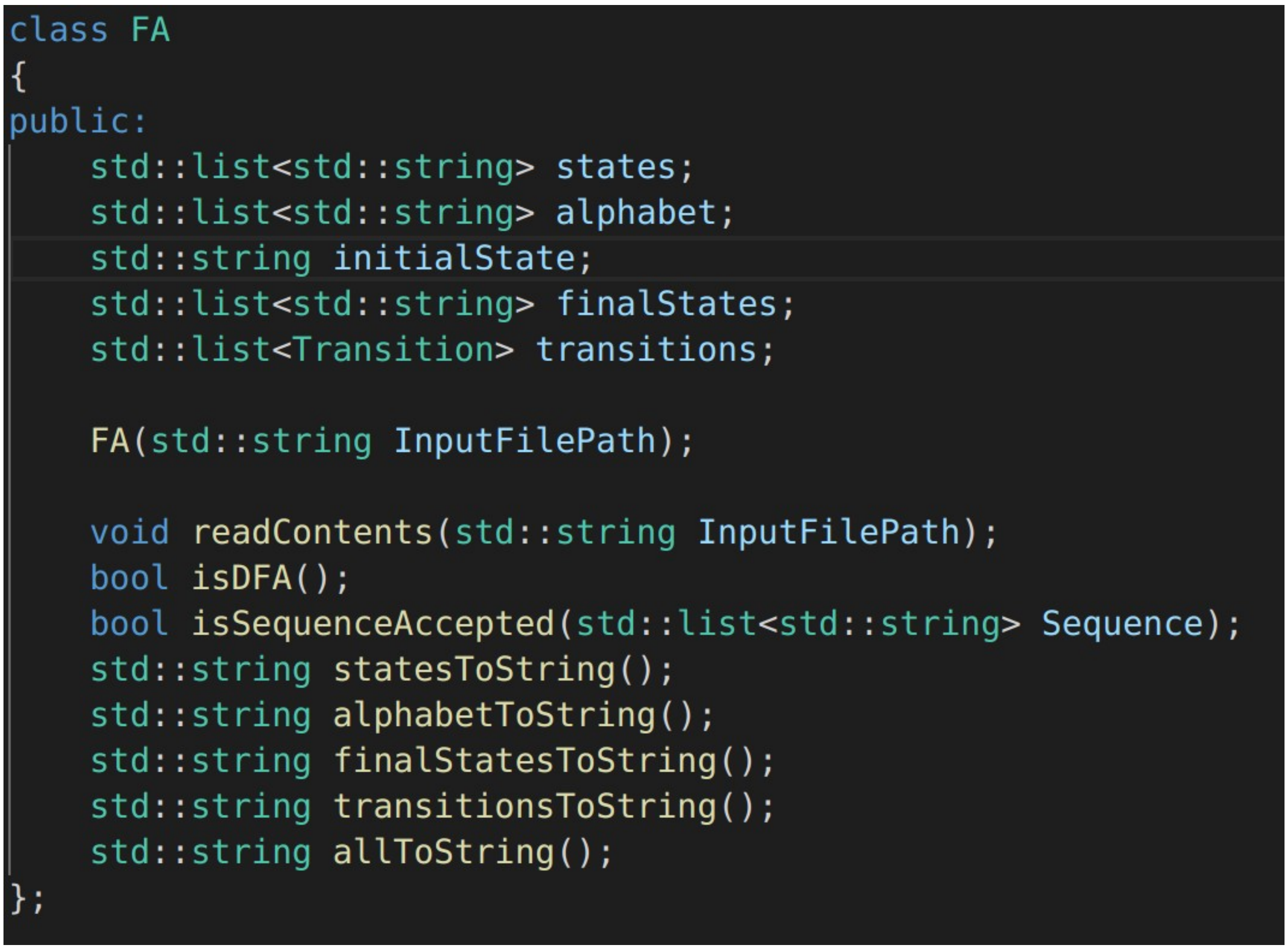
```
class MenuFA
{
public:
    FA* fa;

    MenuFA(FA* Fa);

    void ui();
    void printMenu();
};
```

It takes a pointer to FA as it's attribute.

It provides 2 methods:

printMenu() - prints the menu on the screen once
ui() - it is a loop that prints the menu on the screen, and calls the appropriate
        functions from the FA class, depending on the user input

The FA class handles all operations on the finite automaton. It is defined as follows:

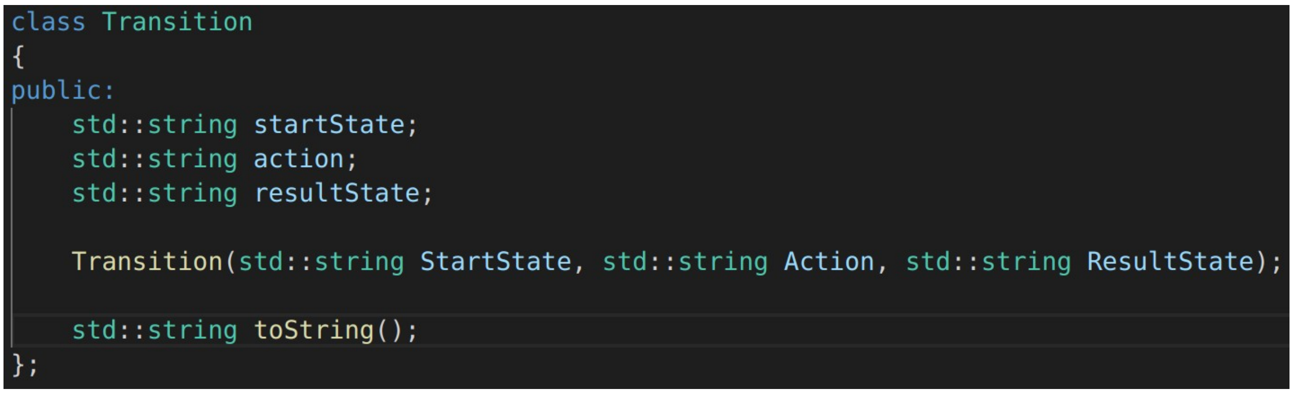


```cpp
class FA
{
public:
    std::list<std::string> states;
    std::list<std::string> alphabet;
    std::string initialState;
    std::list<std::string> finalStates;
    std::list<Transition> transitions;

    FA(std::string InputFilePath);

    void readContents(std::string InputFilePath);
    bool isDFA();
    bool isSequenceAccepted(std::list<std::string> Sequence);
    std::string statesToString();
    std::string alphabetToString();
    std::string finalStatesToString();
    std::string transitionsToString();
    std::string allToString();
};
```

It contains all the states, alphabet, initial state, final states, and transitions.

Methods:
readContents(string InputFilePath) – reads the contents of a FA from an input file and populates it’s attributes
isDFA() - returns true if the FA is deterministic
isSequenceAccepted(list<string> Sequence) – returns true if the provided sequence is accepted by the FA
It also has some “toString” functions, for each attribute.

The Transition class is a simple structure that holds all information about one transition: starting state, action, and resulting state. It is defined as follows:



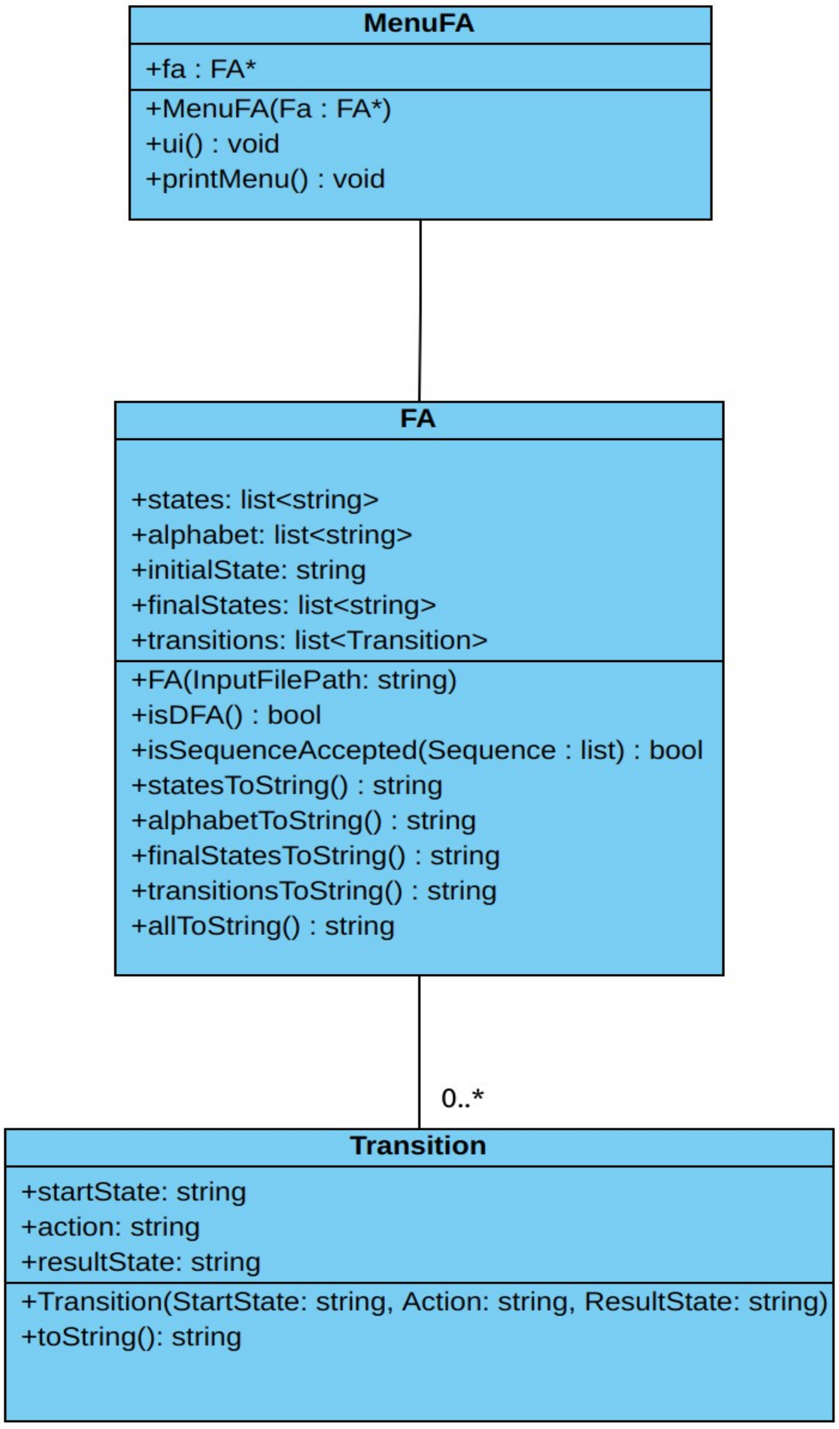
```cpp
class Transition
{
public:
    std::string startState;
    std::string action;
    std::string resultState;

    Transition(std::string StartState, std::string Action, std::string ResultState);

    std::string toString();
};
```

The form in which the FA.in file should be written:

form ::= “States” state{state} “Alphabet” {alphabet} “InitialState” state
               “FinalStates” state{state} “Transitions” {transition}

state ::= “a” | “b” | … | “z”
alphabet ::= “1” | “2” | … | “9”
transition ::= “(“ state “, ” alphabet “) = ” state

Class diagram:


MenuFA
+fa : FA*
+MenuFA(Fa : FA*)
+ui() : void
+printMenu() : void
FA
+states: list<string>
+alphabet: list<string>
+initialState: string
+finalStates: list<string>
+transitions: list<Transition>
+FA(InputFilePath: string)
+isDFA() : bool
+isSequenceAccepted(Sequence : list) : bool
+statesToString() : string
+alphabetToString() : string
+finalStatesToString() : string
+transitionsToString() : string
+allToString() : string
0..*
Transition
+startState: string
+action: string
+resultState: string
+Transition(StartState: string, Action: string, ResultState: string)
+toString(): string