实验一：实验报告

网安2201常兆鑫

## 一、实验要求整理：

1. **DFA输入组成**（需从文本文件中读取）：
   * 字符集（alphabet）
   * 状态集（states）
   * 初始状态（start state）
   * 接受状态集（accepting states）
   * 状态转移表（transition table）
2. **程序功能需求**：
   * 校验：起始状态是否唯一、是否属于状态集；接受状态是否为空、是否属于状态集；
   * 输出：长度不超过N的所有合法字符串；
   * 判断：输入一个字符串（或随机生成）是否是合法的（即是否被DFA接受）。

## 二、实验目的

1. 理解确定性有限自动机的基本原理与五元组结构（状态集、输入符号集、起始状态、接受状态集、状态转移函数）；
2. 掌握用C/C++语言实现DFA模拟器的方法；
3. 能够验证一个字符串是否被某个DFA所接受，并枚举该DFA语言中的合法字符串。

## 三、实验环境

* 操作系统：Linux / debian12
* 编译工具：clang->cmake
* 编程语言：C++

## 四、实验内容与步骤

### 1. 输入DFA五元组（读取自文件）

设计 dfa.txt 文件保存如下DFA结构：

* 字符集（Alphabet）
* 状态集（States）
* 初始状态（Start State）
* 接受状态集（Accept States）
* 状态转移表（Transitions）

文件内容示例如下：

alphabet: a b

states: q0 q1 q2

start: q0

accept: q2

transitions:

q0 a q1

q0 b q0

q1 a q2

q1 b q0

q2 a q2

q2 b q2

### 2. 程序结构设计

程序主要模块如下：

* loadDFA(filename)：读取DFA定义文件并解析五元组
* checkDFAValid()：检查起始状态和接受状态是否合法
* runDFA(input)：模拟字符串运行过程并判断接受与否
* generateAllStrings(N)：枚举DFA中所有长度不超过N的合法字符串

## 五、实验截图与结果分析

### 编译执行截图：

* 成功编译程序
* 正确读取上层目录的 dfa.txt 文件
* 输出符合 DFA 接受的所有合法字符串
* 测试字符串结果准确反映 DFA 语言识别情况

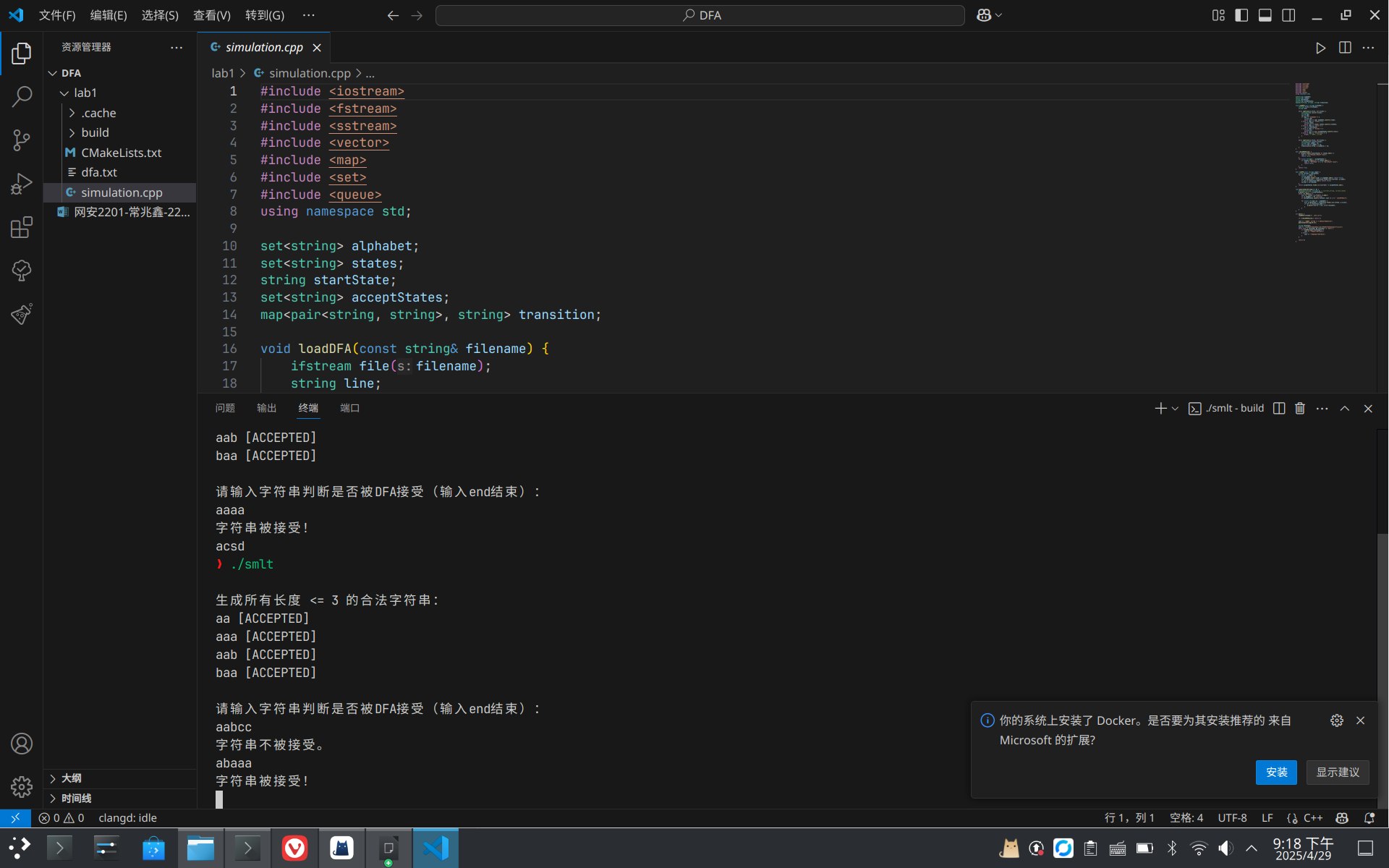
### 结果分析：

程序正确模拟了DFA的运行过程，能：

* 校验DFA结构合法性；
* 正确识别合法/非法字符串；
* 输出语言中所有符合长度限制的字符串。

## 六、实验总结

本次实验通过编程实现了DFA的基本结构和工作机制。加深了对自动机模型的理解，并锻炼了文件读取、数据结构（集合、映射）、搜索算法等编程能力。通过实验也进一步掌握了如何将形式化定义转化为程序逻辑，具备了构建基础语言识别器的能力。



## 七 、实验代码

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110 | #include <iostream>  #include <fstream>  #include <sstream>  #include <vector>  #include <map>  #include <set>  #include <queue>  **using** **namespace** std;    set<string> alphabet;  set<string> states;  string startState;  set<string> acceptStates;  map<pair<string, string>, string> transition;    **void** loadDFA(**const** string& filename) {      ifstream file(filename);      string line;    **while** (getline(file, line)) {          istringstream iss(line);          string key;          iss >> key;  **if** (key == "alphabet:") {              string sym;  **while** (iss >> sym) alphabet.insert(sym);          } **else** **if** (key == "states:") {              string state;  **while** (iss >> state) states.insert(state);          } **else** **if** (key == "start:") {              iss >> startState;          } **else** **if** (key == "accept:") {              string acc;  **while** (iss >> acc) acceptStates.insert(acc);          } **else** **if** (key == "transitions:") {  **break**; // 后面是转移函数          }      }    **while** (getline(file, line)) {          istringstream iss(line);          string from, symbol, to;          iss >> from >> symbol >> to;          transition[{from, symbol}] = to;      }  }    **bool** checkDFAValid() {  **if** (states.find(startState) == states.end()) {          cout << "起始状态不在状态集中！\n";  **return** **false**;      }  **for** (**const** string& s : acceptStates) {  **if** (states.find(s) == states.end()) {              cout << "接受状态" << s << "不在状态集中！\n";  **return** **false**;          }      }  **return** **true**;  }    **bool** runDFA(**const** string& input) {      string current = startState;  **for** (**char** c : input) {          string sym(1, c);  **if** (alphabet.find(sym) == alphabet.end()) **return** **false**;          auto it = transition.find({current, sym});  **if** (it == transition.end()) **return** **false**;          current = it->second;      }  **return** acceptStates.find(current) != acceptStates.end();  }    **void** generateAllStrings(**int** N) {      queue<pair<string, string>> q; // {current\_string, current\_state}      q.push({"", startState});  **while** (!q.empty()) {          auto [s, state] = q.front(); q.pop();  **if** (s.length() > N) **continue**;  **if** (acceptStates.count(state)) cout << s << " [ACCEPTED]\n";    **for** (**const** string& sym : alphabet) {              auto it = transition.find({state, sym});  **if** (it != transition.end()) {                  q.push({s + sym, it->second});              }          }      }  }    **int** main() {      loadDFA("../dfa.txt");    **if** (!checkDFAValid()) **return** 1;        cout << "\n生成所有长度 <= 3 的合法字符串：\n";      generateAllStrings(3);        string testInput;      cout << "\n请输入字符串判断是否被DFA接受（输入end结束）：\n";  **while** (cin >> testInput && testInput != "end") {  **if** (runDFA(testInput)) {              cout << "字符串被接受！\n";          } **else** {              cout << "字符串不被接受。\n";          }      }    **return** 0;  } |