

**形式语言与自动机实验报告**



**题目： NFA到DFA的转化**

**小 组： 第二组**

**班 级： 2022211304**

**学 院： 计算机学院（国家示范性软件学院）**

**2024年 3 月 29 日**

**一、小组成员信息：**

王骊琨：

赵宇鹏：

高天涛：  
曹文卿：

1. **实验目的**

利用编程实现NFA到DFA的转化，理解不同自动机的转化过程。

1. **实验环境描述**

文件夹中存放的是本次实验的源代码文件以及可执行程序，源代码主要采用C++语言编写。

编不出来了

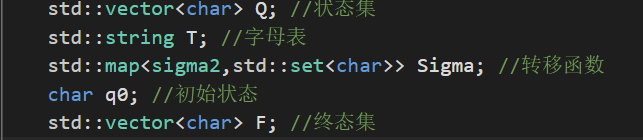
1. **程序设计思路及核心算法**

程序总体结构如下图所示：

#### 

具体程序设计思路与上课所学相同，首先创建一个FA类，用来储存NFA的各状态集合以及转移函数，然后通过循环来实现子集构造法，通过遍历每一个状态集合找到新的状态集合直到没有新的状态集合产生，再根据新的状态集合来生成新的转移函数，最后生成所转换的DFA并输出。

涉及到的状态属性参数如下



对有限状态自动机的各参数存储结构如下



通过sigma2类来储存转移函数，pre表示转换前的状态，input表示字母表输入，这两个参数由Sigma中的first来储存，相应的转换后的状态由second字符数组来储存。

下面开始具体介绍各模块代码及作用：

##### intputNFA模块:

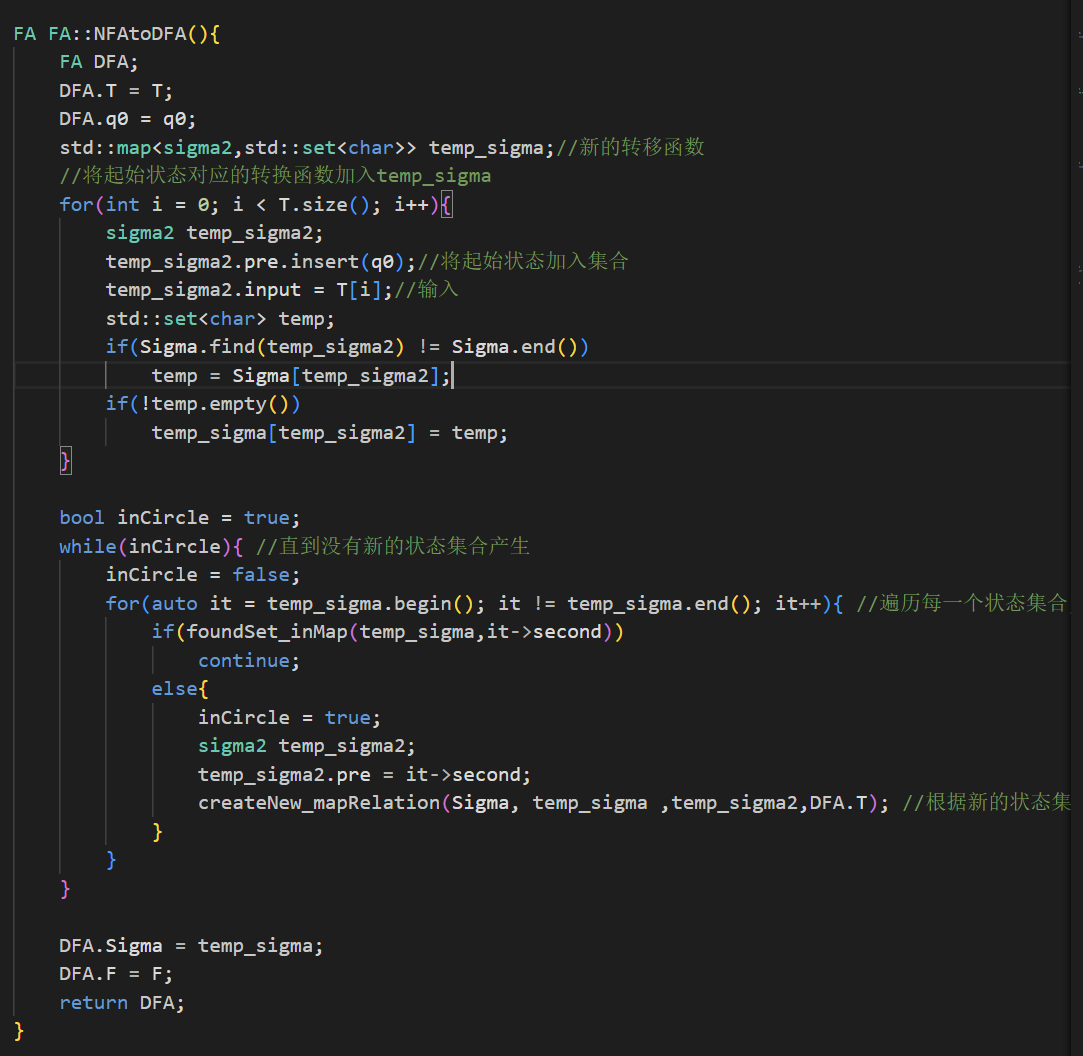
其具体代码如下：



InputNFA函数首先读取NFA的状态集内元素数目，转换函数数目，终态集元素数目来创建对应的状态集合以及转移函数集合，然后按顺序对状态集，字母表，转换函数，初始状态以及终止状态集进行读取和储存从而达到储存NFA的目的。

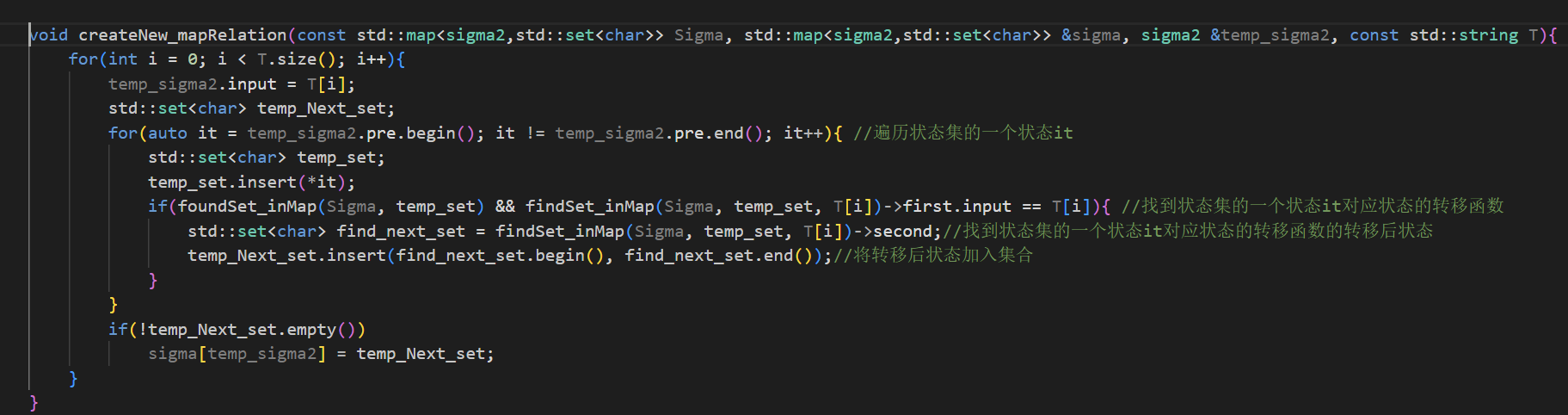
##### NFAtoDAF模块：

代码如下：



如前面所说，NFA转换为DFA的思路为子集构造法，我们直到NFA转DFA字母表以及初始状态是不会改变的，所以直接复制即可，此后，根据子集构造法，我们可以利用temp\_sigma2来先储存初始状态对应不同输入字符的转换函数，若在NFA的转换函数中找到了对应temp\_sigma2的转换函数，则将该转换函数加入到新的转换函数中去，从而完成了子集构造法的初始化，即将初始状态q0对应的转换函数加入到新的转换函数中去，此后对新的转换函数temp\_sigma里面的状态进行遍历，目前即为对初始状态生成的状态进行遍历，若这些状态的生成状态利用foundSet\_inMap函数没有在temp\_sigma中找到的话，那么就是一个新状态集合，即一个新子集，便将其加入到temp\_sigma中去，然后使用createNew\_mapRelation函数来构造新的转移函数，

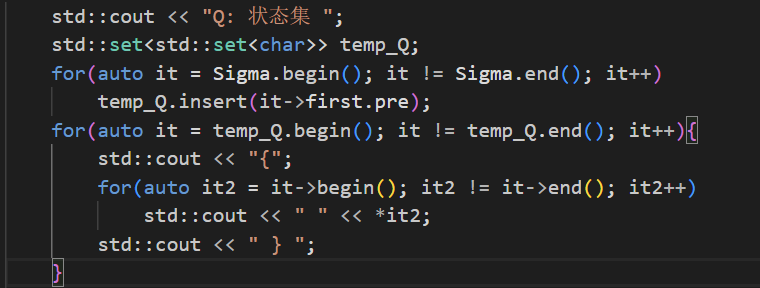
createNew\_mapRelation函数代码如下：

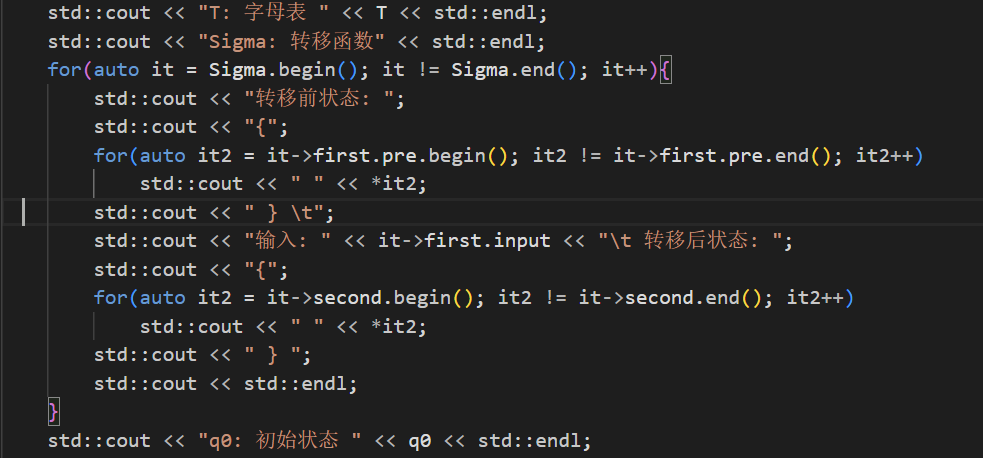


对所有字母表进行遍历输入，来对temp\_sigma中的所有状态进行输入，首先，将当前前驱状态插入临时集合 temp\_set 中。然后，使用 foundSet\_inMap 函数检查是否存在与当前前驱状态对应的转移函数，并且使用 findSet\_inMap 函数查找与当前前驱状态和当前输入字符对应的转移函数条目。如果找到了匹配的转移函数，就获取对应的下一个状态集合，并将其加入临时集合 temp\_Next\_set 中，最后将temp\_Next\_set中的新转换函数存入到temp\_sigma中，最后将temp\_sigma赋给DFA，这样就把新状态以及相应的新转换函数给加入到temp\_sigma中去了，循环执行至没有新状态产生，至此DFA的新转移函数就构造好了，再把NFA的F赋给DFA来表示包含F中元素的状态集合为终止集，从而实现NFA到DFA的转换。

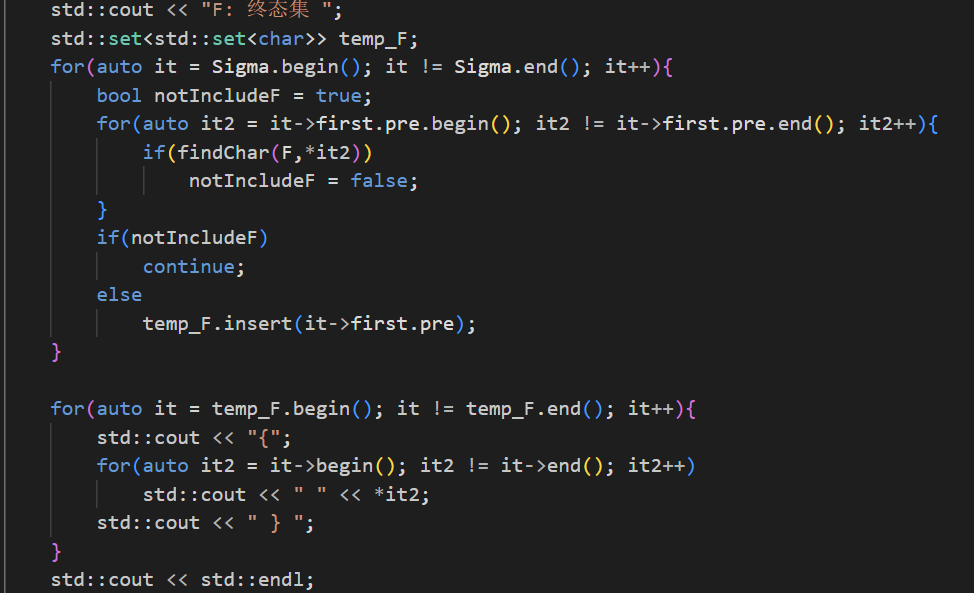
##### outputDFA模块：

首先输出状态集



由于first中存储着转换前状态以及字母表输入，所以状态集的输出只需要将first中的pre集合存入temp\_Q中再输出即可，字母表，转换函数以及初始状态同理

由于在进行NFA到DFA的转换时，DFA的终态集内是NFA的终态集，并未完成转换，所以在输出时应根据F中元素对状态集进行查找，若状态集中有F中元素，则说明其为终态集，进行输出，具体代码如下



1. **测试用例及输出**

**输入格式：**

状态数 转化函数数 终态数

状态集

字母表

转化函数（一行一个）

初始状态

终态集

输入

4 7 2

0123

ab

0a13

0b1

1a2

1b12

2a3

2b0

3b0

0

13

输出

