二、正则表达式→>DFA（1 题，16 分)

1.我们知道一个单词的正则表达式可以转换为 DFA,接着就可以根据该 DFA 图写出实现该单词的词法分析程序。现请写出将 DFA 图转换为词法分析的转换程序【提示:你需要先写出 DFA 图的存储结构，接着才写出基于该结构下的转换程序。】

(1)存储结构

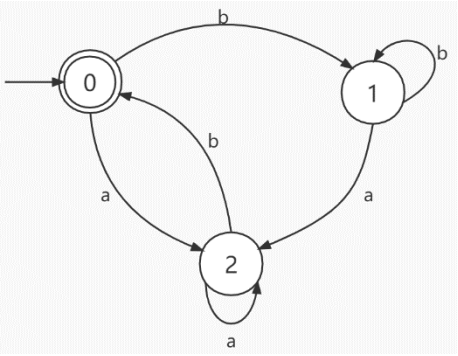
```
1.  //DFA 节点数组以及数量
2.  int dfaNodeNum = 0;                    //DFA 节点总数
3.  //DFA 结构体
4.  struct DFA
5.  {
6.      int startState = 0;                //DFA 的初态
7.      set<int> endStates;                //DFA 的终态集合
8.      set<char> terminator;              //DFA 的终结符集
9.      int trans[100][26] = {};           //DFA 的转移矩阵,行表示状态，列表示终结符
10. };
```

(2)转换程序:

```
1.  string dfaToCode(DFA d)
2.  {
3.      int i, sta;
4.      set<char>::iterator it;
5.      set<int>::iterator it2;
6.      string code = "";                                   //输出的代码
7.       //头文件和变量、初始状态和 while、switch 结构
8.      code += "#include<iostream>\n#include<string.h>\n\nusing namespace std;\\n\n
    int main()\n{\n\tstring str;\n\tint i = 0;\n\tcin >> str;\n\tint length = str.si
    ze();\n" + "\tint state = " + to_string(d.startState) + ";\n\twhile (state != -
    1 && i < length)\n\t{\n\t\tswitch (state)\n\t\t{\n";
9.      for (i = 0; i < dfaNodeNum; i++)
10.     {
11.         code += "\t\tcase " + to_string(i) + ":\n";     //case 结构
12.         bool haveEdge = false;                          //节点是否有出边的标志
13.         for (it = d.terminator.begin(); it != d.terminator.end(); it++)
14.         {
15.             sta = d.trans[i][*it - 'a'];
16.             if (sta != -1)                              //有出边
17.             {
18.                 //第一条边用 if、不是第一条边则用 else if
19.                 if(haveEdge) code += "\t\t\telse if (str[i] == '";
20.                 else code += "\t\t\tif (str[i] == '";
21.                 code += *it;                            //终结符
22.                 code += "')\n\t\t\t{\n\t\t\t\tstate = " + to_string(sta) + ";\n\
    t\t\t}\n";                                              //转换的状态
```

```
                haveEdge = true;
            }
        }
        //节点有出边则加 else，没有出边则要加入是否是最后一个字符的判断（防止死循环）
        if (haveEdge)
            code += "\t\t\telse state = -1;\n";
        else
            code += "\t\t\tif (i < length)\n\t\t\t\tstate = -1;\n";
        code += "\t\t\ti++;\n\t\t\tbreak;\n";          //读取下一个字符、break
    }
    code += "\t\tdefault:\n\t\t\tbreak;\n\t\t}\n\t}\n\tif (";
    //switch、while 结构结束
    it2 = d.endStates.begin();                        //遍历终态集
    code += "state == " + to_string(*it2++);          //加入第一个终态
    while (it2 != d.endStates.end())
    {
        code += " || state == " + to_string(*it2);//如果有更多终态则加入或运算符
        it2++;
    }
    code += ")\n\t\tcout << \"accept\" << endl;\n\telse\n\t\tcout << \"error\" << endl;\n\treturn 0;\n}\n";                                //输出和 return
    return code;
}
```

示例：
(a|b)*ab 的最小化 DFA 为：



生成词法分析程序为：

```
#include<iostream>
#include<string.h>

using namespace std;

int main()
{
    string str;
```

```
    int i = 0;
    cin >> str;
    int length = str.size();
    int state = 1;
    while (state != -1 && i < length)
    {
        switch (state)
        {
        case 0:
            if (str[i] == 'a')
            {
                state = 2;
            }
            else if (str[i] == 'b')
            {
                state = 1;
            }
            else state = -1;
            i++;
            break;
        case 1:
            if (str[i] == 'a')
            {
                state = 2;
            }
            else if (str[i] == 'b')
            {
                state = 1;
            }
            else state = -1;
            i++;
            break;
        case 2:
            if (str[i] == 'a')
            {
                state = 2;
            }
            else if (str[i] == 'b')
            {
                state = 0;
            }
            else state = -1;
            i++;
            break;
```

```
53.         default:
54.             break;
55.         }
56.     }
57.     if (state == 0)
58.         cout << "accept" << endl;
59.     else
60.         cout << "error" << endl;
61.     return 0;
62. }
63.
```