華南師範大學

本科学生实验（实践）报告

院　　系：计 算 机 学 院

实验课程：编译原理项目

实验项目：编译原理项目

指导老师：黄煜廉

开课时间：2023 ～ 2024 年度第 2 学期

专　　业：计算机科学与技术

班　　级：计科 1 班

学　　生：李达良

学　　号：20203231004

华南师范大学教务处

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

一、项目内容

第一个项目任务内容：

（1）以文本文件的方式输入某一高级程序设计语言的所有单词对应的正则表达式，系统需要提供一个操作界面，让用户打开某一语言的所有单词对应正则表达式文本文件，该文本文件的具体格式可根据自己实际的需要进行定义。

（2）需要提供窗口以便用户可以查看转换得到的 NFA（可用状态转换表呈现）

（3）需要提供窗口以便用户可以查看转换得到的 DFA（可用状态转换表呈现）

（4）需要提供窗口以便用户可以查看转换得到的最小化 DFA（可用状态转换表呈现）

（5）需要提供窗口以便用户可以查看转换得到的词法分析程序（该分析程序需要用 C 语言描述）[只能使用讲稿中的方法一或方法二来生成词法分析程序]

（6）对要求(5)得到的源程序进行编译生成一个可执行程序，并以该高级程序设计语言的一个源程序进行测试，输出该该源程序的单词编码。需要提供窗口以便用户可以查看该单词编码。

（7）对系统进行测试: （A）先以 TINY 语言的所有单词的正则表达式作为文本来测试，生成一个 TINY 语言的词法分析源程序；（B）接着对这个词法分析源程序利用 C/C++编译器进行编译，并生成可执行程序；（C）以 sample.tny 来测试，输出该 TINY 语言源程序的单词编码文件 sample.lex。

（8）要求应用程序为 Windows 界面

（9）书写完善的软件文档

第二个项目任务内容：

（1）以文本文件的方式输入某一高级程序设计语言的所有语法对应的 BNF 文法，因此系统需要提供一个操作界面，让用户打开某一语言的所有语法对应的 BNF 文法的文本文件，该文本文件的具体格式可根据自己实际的需要进行定义。

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

（2）求出文法的每个非终结符号的 First 集合和 Follow 集合，并需要提供窗口以便用户可以查看该结果（可用两张表格的形式分别进行呈现）

(3)需要提供窗口以便用户可以查看文法对应的 LR(0)DFA 图。（可以用画图的方式呈现，也可用表格方式呈现该图点与边数据）

（4）构造出 SLR(1)分析表，并需要提供窗口以便用户可以查看该结果（可用表格形式进行呈现）

（5）采用 SLR(1)语法分析方法进行语法分析并生成相应的语法树，每个语句的语法树结构可根据实际的需要进行定义。（语法树需要采用树状形式进行呈现）

（6）以 TINY 语言的所有语法以及第一项任务的测试结果 sample.lex 作为测试，并生成对应的语法树并呈现出来。

（7）要求应用程序为 Windows 界面

（8）书写完善的软件文档

（9）选做内容：可以生成某种中间代码【具体的中间代码形式可以自定】。

第三个项目任务要求

mini-c 语言作为测试

（1）以 mini-c 的词法进行测试，并以至少一个 mini-c 源程序进行词法分析的测试（该 mini-c 源程序需要自己根据 mini-c 词法和语法编写出来，类似于 sample.tny）。

（2）以 mini-c 的语法进行测试，并以测试步骤（1）的源程序所生成的单词编码文件进行语法分析，生成对应的语法树。

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

二、项目目的

第一个项目任务目的：

在这个项目中，我们将深入探讨程序编译过程中的重要步骤，并着手模仿实现一个编译器的关键功能。具体而言，项目的主要目标是开发一个词法分析程序生成器。这个生成器将能够解析特定格式的文件，这些文件包含了一个编程语言中各种词汇类型的正则表达式描述。通过分析这些表达式，生成器将自动构建一个词法分析器，该分析器能够识别并分类源代码中的词汇元素，如标识符、关键字、符号等。通过实际应用生成的词法分析器，我们可以有效地提高编译器的编译效率和准确性，为更深入的语法分析和语义处理奠定基础。

第二个项目任务目的：

在这个项目中，我们将探索和实现一个高级编程语言的语法分析器，通过精确地处理 BNF 文法，实现语法的自动分析。本项目的主要目的是通过开发一个软件系统，该系统能够输入、解析和展示高级编程语言的 BNF 文法描述，并基于此自动生成 First 集和 Follow 集，构建 LR(0)DFA，生成 SLR(1)分析表，执行 SLR(1)语法分析，并最终构建语法树。该系统将提供一个用户友好的 Windows 操作界面，使用户能够方便地上传文法文件、查看各种分析结果，并通过图形化界面展示语法树和 LR(0)DFA 图、SLR1 分析表。此外，系统还将支持以 TINY 语言、MINIC 语言的 BNF 文法和词法分析结果为基础进行测试，验证分析器的正确性和效能。通过这个项目，我们旨在深化对编译原理的理解，提高编译过程的自动化程度，并为可能的中间代码生成奠定基础。

第三个项目任务目的：

测试前两个项目任务的完成出来的编译器的完成度和准确性。

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 **学　号** 20203231004

专　　业 计算机科学与技术 **年级、班级** 2021 级 1 班

课程名称 编译原理项目 **实验项目** 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 **实验评分**

三、项目文档：

（一）项目任务一：

1. 系统设计

1.1 系统的关键数据结构

（1）NFA 数据结构

数据结构名称	nfaEdge	
数据结构类型	结构体	
数据结构用途	Nfa 图的边	
字段名称	字段类型	说明
c	char	这个边的字符
next	nfaNode*	这个边指向的下一个结点

数据结构名称	nfaNode	
数据结构类型	结构体	
数据结构用途	Nfa 图的结点	
字段名称	字段类型	说明
id	int	结点的唯一编号
isStart	bool	初态标识
isEnd	bool	终态标识
edges	vector<nfaEdge>	边集合

数据结构名称	NFA
数据结构类型	结构体
数据结构用途	Nfa 图

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

字段名称	字段类型	说明
start	nfaNode*	起始点
end	nfaNode*	终点

（2）DFA 数据结构

数据结构名称	dfaNode	
数据结构类型	结构体	
数据结构用途	dfa 图的结点	
字段名称	字段类型	说明
flag	string	是否包含终态（+）或初态（-）
nfaStates	set<int>	该 DFA 状态包含的 NFA 状态的集合
transitions	map<char, set<int>>	字符到下一状态的映射

数据结构名称	dfaStatusSet
数据结构类型	set<set<int>>
数据结构用途	DFA 状态去重的结果集

数据结构名称	dfaTable
数据结构类型	vector<dfaNode>
数据结构用途	DFA 图最终的结果

（3）最小化 DFA 数据结构

数据结构名称	dfaMinNode

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

数据结构类型	结构体	
数据结构用途	最小化 dfa 图结点	
字段名称	字段类型	说明
flag	string	是否包含终态（+）或初态（-）
id	int	序号
transitions	map<char, set<int>>	字符到下一状态的映射

数据结构名称	dfaMinTable
数据结构类型	vector<dfaMinNode>
数据结构用途	最小化 DFA 图最终的结果

1.2 系统功能结构

在进行项目设计时，我将项目任务一分成了两个大模块：图形化操作模块和数据处理模块。

其中图形化操作模块分为三个部分：输入模块、展示模块、lex 文件展示模块；数据处理模块分为五部分：正则表达式处理模块、NFA 处理模块、DFA 处理模块、DFA 最小化处理模块、词法分析程序生成模块。

具体如图所示：

华 南 师 范 大 学 实 验 报 告

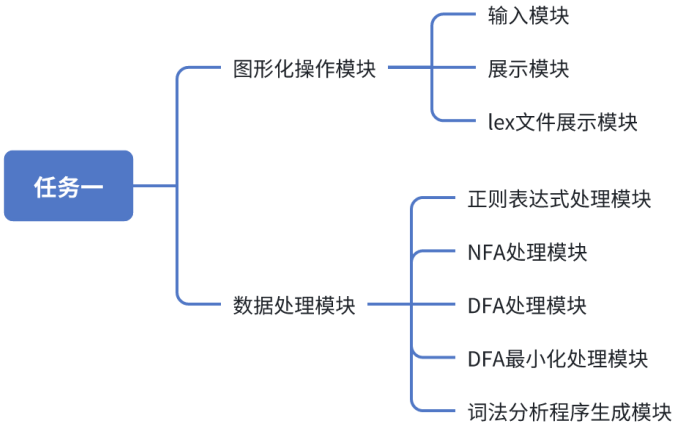
学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



2. 功能模块设计及实现

2.1 图形化操作模块

（1）描述

通过 QT 实现 UI 的设计，给予一个正则表达式输入框，输入框旁边有四个按钮，分别是上传正则表达式、下载正则表达式、是否词法分析忽略大小写和查看规则，方便使用者的输入和保存。填写正则表达式后，点击开始分析，分析成功后，点击“NFA”、“DFA”、“DFA 最小化”、“词法分析程序”就可以获得对应的结果，操作简单便捷。

使用者点击“词法分析程序”按钮，选择对应的路径，生成词法分析程序代码，通过编译运行词法分析程序的代码，即可获得对应单词编码文件“output.lex”，我们提供“查看 lex 文件”按钮，使得可以读取对应的 lex 文件。

（2）QT 设计图

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

（3）界面截图

2.2 正则表达式处理模块

（1）描述

1. 我们规定正则表达式输入格式如下：

* 第一行：keyword，多个用 | 进行拼接成正则表达式
* 第二行：除了注释符号的专用符号的正则表达式，用|连接，如果与正则表达

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

式符号冲突（包括：+、|、（、）、*、？、[、]、~），用\进行包围，如：\+\

* 第三行：第二行符号对应的名称，用|分隔

* 第四行：标识符的正则表达式，可以用\letter\表示所有字母，\num\表示所有数字

* 第五行：数字的正则表达式

* 第六行：注释的正则表达式，用~表示除结束符号的任意字符，如{~}

2. 我们进行符号替换：

使用一个映射替换正则表达式中的特定符号。由于正则表达式中如|、？、+等符号会对解析正则表达式造成影响，我们规定，如果使用这些符号需要对其加入\处理，如：\+\。但是我们的正则表达式一般是按字符来进行解析，所以为了避免字符冲突，我们使用了 ascii 码中的不可打印字符进行替换，以此来规避对应的影响。

3. 进行分割处理：

我会对关键词、符号、符号名称、注释进行分割，并保存。

将修改后的字符串按行分割为 QStringList 对象 lines。

如果 lines 不恰好为 6 行，则返回错误消息。

4. 正则表达式后处理：

afterHandleRegex(output)对最终的正则表达式进行进一步处理，主要是处理[] + 和添加连接符。比如[]其实就是把里面的元素用或运算符进行拼接，正闭包就是转换成形如 ss*的形式。

（2）测试示例

tiny 的正则表达式，经过处理后，对应的输入文件如下所示：

```
if|then|else|end|repeat|until|read|write
\+\ | - | \*\ | / | % | < | <> | <= | >= | > | = | ; | := | \(\ | \)\
PLUS | MINUS | MULTIPLY | DIVIDE | MOD | LT| NE | LTEQ | RTEQ | RT | EQ | SEMI | ASSIGN | LPAN | RPAN
letter(letter|num)*
num num*
{~}
```

（3）核心算法

```
// 下面map防止字符冲突
// 符号->字符串map
map<char, string> m1 = { {(char)1,"num"}, {(char)2, "letter"},{(char)3, "\\+\\"},{(char)4, "\\|\\"},{(char)5, "\\(\\"},{(char)6, "\\)\\"},{(char)7, "\\*\\"},{(char)16, "\\?\\"},{(char)17, "\\[\\"},{(char)18, "\\]\\"},{ (char)19,"\\~\\"}, {(char)20,"\\n"}};
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
// 字符串->符号map
map<string, char> m2 = { {"num",(char)1}, {"letter", (char)2 },{"\\+\\",
(char)3 },{"\\|\\", (char)4 },{ "\\(\\",
(char)5},{ "\\)\\" ,(char)6},{"\\*\\" ,(char)7},{ "\\?\\",(char)16},{"\\[\\",
(char)17},{"\\]\\", (char)18 }, {"\\~\\", (char)19},{"\\n",(char)20} };

string handleAllRegex(QString allRegex, bool isLowerCase) {

    // 不区分大小写的话，全部转为小写
    if (isLowerCase) {
        allRegex = allRegex.toLower();
    }
    // 去除空格
    allRegex.replace(" ","");
    // 使用map映射关系替换正则表达式中的符号
    for (const auto& pair : m2) {
        allRegex.replace(QString::fromStdString(pair.first), QString(pair.second));
    }

    // 将文本内容按行分割成一个字符串列表
    QStringList lines = allRegex.split("\n");

    if (lines.size() != 6) {
        return "输入的正则表达式不符合规范，不为6行，请检查！";
    }
    getKeyWords(lines[0]);  // 获取关键词
    string res = getCommentSymbol(lines[5]);  // 获取注释开始符号和结束符号后
    res = getOpName(lines[1], lines[2]);


    // 拼接生成NFA和DFA图的正则表达式
    QString output;
    for (int i = 1; i < 6; ++i) {
        if (i == 2) continue;   // 第三行不做处理
        output += "(" + lines[i] + ")";
        if (i < 5) {
            output += "|";
        }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
    }
    finalRegex = output.toStdString();
    qDebug() <<("拼接后的正则表达式：") << QString::fromStdString(finalRegex);

    finalRegex = afterHandleRegex(output).toStdString();
    qDebug() << ("处理后的正则表达式：") << QString::fromStdString(finalRegex);

    return res;
}
```

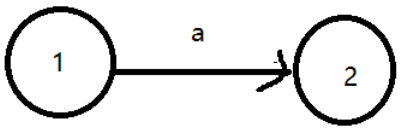
2.3 NFA 生成

2.3.1 描述

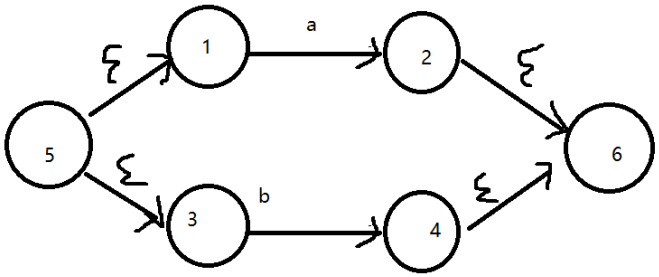
（1） 基本 NFA 图的生成

对于正则表达式转 NFA 图，其实我们可以参考逆波兰表达式的双栈法，对正则表达式进行一个扫描，用两个栈，一个栈存入运算符，一个栈存 NFA 图，定义好正则表达式运算符的优先级。

当我们遇到一个字符的时候，我们就要创建一个 NFA 图，如下图，同时初态和终态应该都为 true：



然后，假如我们遇到一个运算符，我们得判断优先级来进行处理，以或运算为例子：a|b，我们事先一定是存入了 a 和 b 的 NFA 图到 NFA 栈中，pop 出来后，我们要对他们进行连接，就是要创建一个开始节点和结束节点，并用一个 ε 边连接



其他运算符也是同理：

闭包：

华 南 师 范 大 学 实 验 报 告

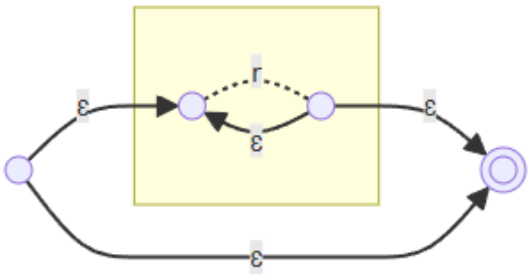
学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

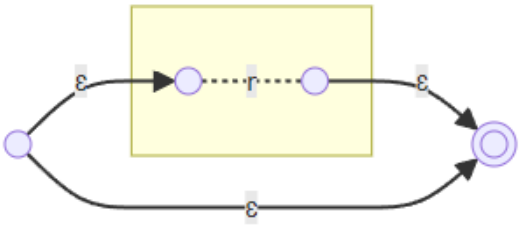
课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



可选：



（2） 优先级的处理

类似于平时的数学运算法则，正则表达式的符号也有结合的优先级，我们将正则表达式转换为 NFA 的最后一步就要解决该问题。

优先级（数字越大越低）	符号	含义	实际编码（数字越大优先级越高）
1	（Regex）	正则表达式	/
2	* ？+	单目运算符	3
3	&	连接运算	2
4	|	选择	1
5	（	左括号	0

（3） NFA 输出

采用双栈法后，我们会得到一个完整的 NFA 图，但我们要将其转换成状态转换表，所以，我们要用 DFS 深搜，得到每个节点在每个字符下的一个状态转换，方便我们后续的处理。

前面我提到过，我会将一些特殊字符进行转变成不可打印字符处理，因此我们在输出的时候就要着重处理这一部门。

2.3.2 单元测试

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.3.3 核心算法代码

```
/*
* @brief 生成状态转换表
* 使用DFS算法
*/
void createNFAStatusTable(NFA& nfa)
{
    stack<nfaNode*> nfaStack;
    set<nfaNode*> visitedNodes;

    // 初态
    nfaNode* startNode = nfa.start;
    statusTableNode startStatusNode;
    startStatusNode.flag = '-'; // -表示初态
    startStatusNode.id = startNode->id;
    statusTable[startNode->id] = startStatusNode;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
insertionOrder.push_back(startNode->id);
startNFAstatus.insert(startNode->id);

nfaStack.push(startNode);

while (!nfaStack.empty()) {
    nfaNode* currentNode = nfaStack.top();
    nfaStack.pop();
    visitedNodes.insert(currentNode);

    for (nfaEdge edge : currentNode->edges) {
        char transitionChar = edge.c;
        nfaNode* nextNode = edge.next;

        // 记录状态转换信息
        statusTable[currentNode->id].m[transitionChar].insert(nextNode->id);

        // 如果下一个状态未被访问，将其加入堆栈
        if (visitedNodes.find(nextNode) == visitedNodes.end()) {
            nfaStack.push(nextNode);

            // 记录状态信息
            statusTableNode nextStatus;
            nextStatus.id = nextNode->id;
            if (nextNode->isStart) {
                nextStatus.flag += '-'; // -表示初态
                startNFAstatus.insert(nextStatus.id);
            }
            else if (nextNode->isEnd) {
                nextStatus.flag += '+'; // +表示终态
                endNFAstatus.insert(nextStatus.id);
            }
            statusTable[nextNode->id] = nextStatus;
            // 记录插入顺序（排除终态）
            if (!nextNode->isEnd)
            {
                insertionOrder.push_back(nextNode->id);
            }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
            }
        }
    }

    // 顺序表才插入终态
    nfaNode* endNode = nfa.end;
    insertionOrder.push_back(endNode->id);
}
```

2.4 DFA 生成

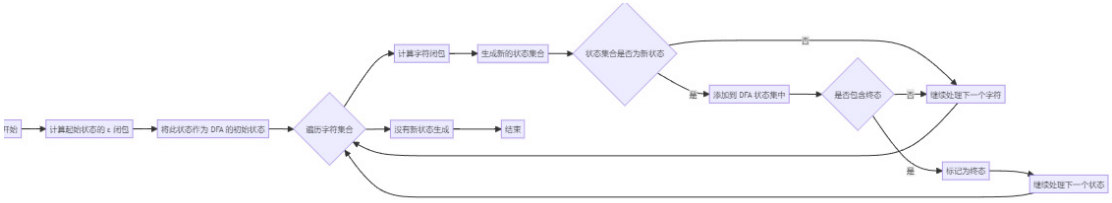
2.4.1 描述

对于 DFA 图，我们已经得到了 NFA 的状态表，我们其实可以很容易得到一个状态的 ε 闭包和字符闭包。从 NFA 的起始状态开始，计算其 ε 闭包，并将这个状态作为 DFA 的初始状态。然后，对每个字符集合（DFA 字符集）进行遍历，计算字符闭包，生成新的状态集合，并建立字符到下一状态的映射。如果生成的状态集合是新的，则将其添加到 DFA 状态集中，同时判断是否包含终态，然后将该状态继续作为下一个状态进行处理，直到没有新状态生成。

在判断有没有新状态生成，其实我们将所有状态放入一个 set 中，去重，看看这个 size 有没有发生变化，如果没有发生变化，就是没有新状态产生。

整个 NFA 到 DFA 的转换过程遵循子集构造法，它通过计算 ε 闭包和字符闭包来确定 DFA 的状态转移。最终，得到了一个表示等价 DFA 的 dfaTable，其中包含了 DFA 节点的信息。

流程图如下：



2.4.2 单元测试

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.4.3 核心算法代码

```
void NFA2DFA(NFA& nfa)
{
    int dfaStatusCount = 1;
    auto start = nfa.start; // 获得NFA图的起始位置
    auto startId = start->id;   // 获得起始编号
    dfaNode startDFANode;
    startDFANode.nfaStates = epsilonClosure(startId); // 初始闭包
    startDFANode.flag = setHasStartOrEnd(startDFANode.nfaStates); // 判断初态终态
    deque<set<int>> newStatus{};
    dfa2numberMap[startDFANode.nfaStates] = dfaStatusCount;
    startStaus = dfaStatusCount;
    if (setHasStartOrEnd(startDFANode.nfaStates).find("+") != string::npos) {
        dfaEndStatusSet.insert(dfaStatusCount++);
    }
    else
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
    {
        dfaNotEndStatusSet.insert(dfaStatusCount++);
    }
    // 对每个字符进行遍历
    for (auto ch : dfaCharSet)
    {
        set<int> thisChClosure{};
        for (auto c : startDFANode.nfaStates)
        {
            set<int> tmp = otherCharClosure(c, ch);
            thisChClosure.insert(tmp.begin(), tmp.end());
        }
        if (thisChClosure.empty())  // 如果这个闭包是空集没必要继续下去了
        {
            continue;
        }
        int presize = dfaStatusSet.size();
        dfaStatusSet.insert(thisChClosure);
        int lastsize = dfaStatusSet.size();
        // 不管一不一样都是该节点这个字符的状态
        startDFANode.transitions[ch] = thisChClosure;
        // 如果大小不一样，证明是新状态
        if (lastsize > presize)
        {
            dfa2numberMap[thisChClosure] = dfaStatusCount;
            newStatus.push_back(thisChClosure);
            if (setHasStartOrEnd(thisChClosure).find("+") != string::npos) {
                dfaEndStatusSet.insert(dfaStatusCount++);
            }
            else
            {
                dfaNotEndStatusSet.insert(dfaStatusCount++);
            }
        }
    }
    dfaTable.push_back(startDFANode);
    // 对后面的新状态进行不停遍历
    while (!newStatus.empty())
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
{
    // 拿出一个新状态
    set<int> ns = newStatus.front();
    newStatus.pop_front();
    dfaNode DFANode;
    DFANode.nfaStates = ns;  // 该节点状态集合
    DFANode.flag = setHasStartOrEnd(ns);
    for (auto ch : dfaCharSet)
    {
        set<int> thisChClosure{};
        for (auto c : ns)
        {
            set<int> tmp = otherCharClosure(c, ch);
            thisChClosure.insert(tmp.begin(), tmp.end());
        }
        if (thisChClosure.empty())  // 如果这个闭包是空集没必要继续下去了
        {
            continue;
        }
        int presize = dfaStatusSet.size();
        dfaStatusSet.insert(thisChClosure);
        int lastsize = dfaStatusSet.size();
        // 不管一不一样都是该节点这个字符的状态
        DFANode.transitions[ch] = thisChClosure;
        // 如果大小不一样，证明是新状态
        if (lastsize > presize)
        {
            dfa2numberMap[thisChClosure] = dfaStatusCount;
            newStatus.push_back(thisChClosure);
            if (setHasStartOrEnd(thisChClosure).find("+") != string::npos) {
                dfaEndStatusSet.insert(dfaStatusCount++);
            }
            else
            {
                dfaNotEndStatusSet.insert(dfaStatusCount++);
            }
        }
    }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
            dfaTable.push_back(DFANode);
        }
    }
```

2.5 最小化 DFA 生成

2.5.1 描述

设置一个分割函数，该函数用于根据字符 ch 将状态集合 node 分成两个子集合。它通过遍历状态集合中的状态，根据字符 ch 找到下一个状态，然后根据下一个状态的映射来决定是否分割成新的状态集合。分割后，删除需要删除的元素，将新的状态集合加入到 vector 的末尾中，实现 DFA 状态的最小化，同时在 dfaMinMap 中更新状态到下标的映射。

在调用这个分割函数的核心算法中，执行以下步骤：

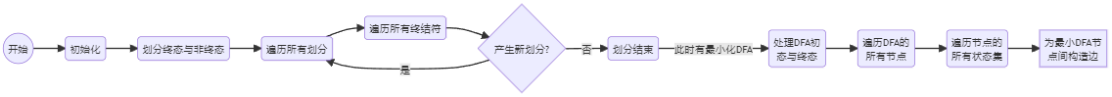
a. 初始化 divideVector 和 dfaMinMap，并将非终态和终态集合添加到 divideVector 中，初始化状态到下标的映射。

b. 进入循环，直到不再有新的状态分割。在每次循环中，遍历 divideVector 中的每个状态集合，然后逐个字符尝试分割状态集合。分割时使用 splitSet 函数。

c. 最小化过程中，会不断地合并相同状态，直到不再有新的状态分割。这个过程确保了生成最小化的 DFA。

d. 创建最小化的 DFA 节点 dfaMinNode，并添加到 dfaMinTable 中。

流程图如下：



2.5.2 单元测试

华 南 师 范 大 学 实 验 报 告

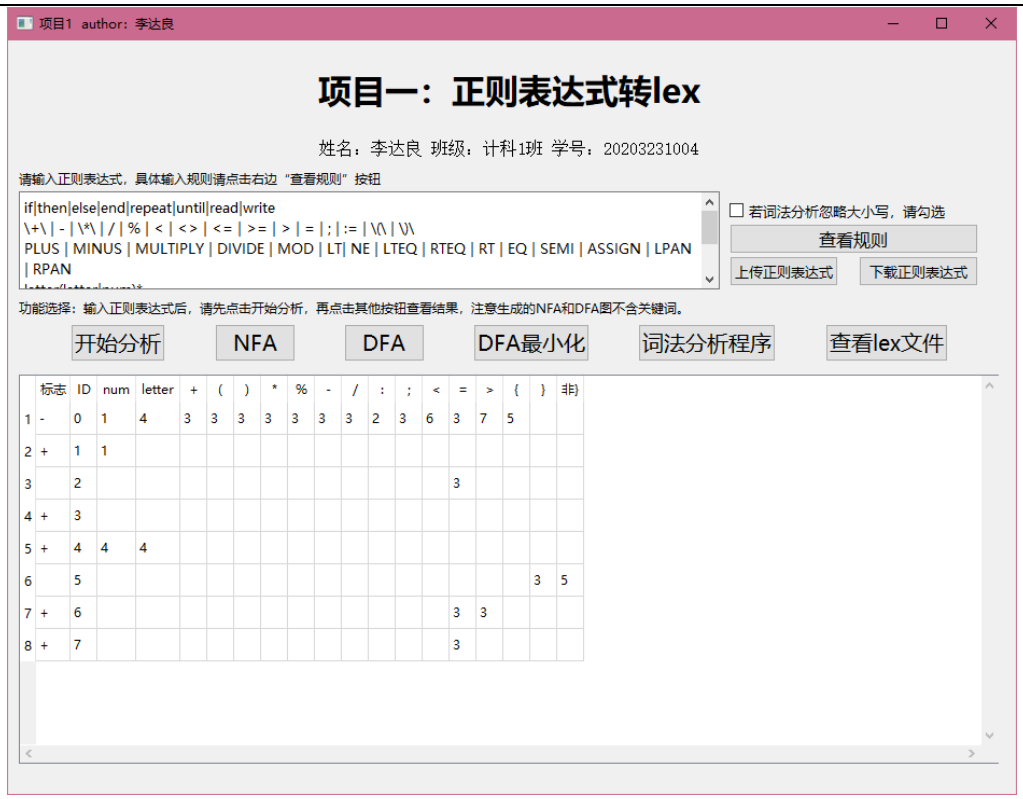
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



	标志	ID	num	letter	+	(	)	*	%	-	/	:	;	<	=	>	{	}	非
1	-	0	1	4	3	3	3	3	3	3	3	2	3	6	3	7	5		
2	+	1	1																
3		2													3				
4	+	3																	
5	+	4	4	4															
6		5																3	5
7	+	6													3	3			
8	+	7													3				

2.5.3 核心算法代码

```
void DFAminimize()
{
    divideVector.clear();
    dfaMinMap.clear();
    // 存入非终态、终态集合
    if (dfaNotEndStatusSet.size() != 0)
    {
        divideVector.push_back(dfaNotEndStatusSet);
    }
    // 初始化map
    for (auto t : dfaNotEndStatusSet)
    {
        dfaMinMap[t] = divideVector.size() - 1;
    }
    divideVector.push_back(dfaEndStatusSet);
    for (auto t : dfaEndStatusSet)
    {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
    dfaMinMap[t] = divideVector.size() - 1;
}
// 当flag为1时，一直循环
int continueFlag = 1;
while (continueFlag)
{
    continueFlag = 0;
    int size1 = divideVector.size();
    for (int i = 0; i < size1; i++)
    {
        // 逐个字符尝试分割状态集合
        for (char ch : dfaCharSet)
        {
            splitSet(i, ch);
        }
    }
    int size2 = divideVector.size();
    if (size2 > size1)
    {
        continueFlag = 1;
    }
}
for (int dfaMinCount = 0; dfaMinCount < divideVector.size(); dfaMinCount++)
{
    auto& v = divideVector[dfaMinCount];
    dfaMinNode d;
    d.flag = minSetHasStartOrEnd(v);
    d.id = dfaMinCount;
    // 逐个字符
    for (char ch : dfaCharSet)
    {
        if (v.size() == 0)
        {
            d.transitions[ch] = -1;   // 空集特殊判断
            continue;
        }
        int i = *(v.begin()); // 拿一个出来
        if (dfaTable[i - 1].transitions.find(ch) == dfaTable[i -
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
1].transitions.end())
            {
                d.transitions[ch] = -1;   // 空集特殊判断
                continue;
            }
            int next_state = dfa2numberMap[dfaTable[i - 1].transitions[ch]];
            int thisNum = dfaMinMap[next_state];    // 这个状态下标
            d.transitions[ch] = thisNum;
        }
        dfaMinTable.push_back(d);
    }
}
```

2.6 词法生成程序代码生成

2.6.1 描述

最小化 DFA 后，我们可以直接利用这个 DFA 构造词法分析程序的代码。我采用字符串拼接的方式进行。

首先将词法分析程序的 C 代码头文件以及关键词数组、操作符以及对应名称的映射关系先写好。在这一部分中，由于 C 代码没有 map 等 STL 库，因此我首先了一个简单的 getValue 函数和 find 函数。

然后确定好哪些情况是需要用代码循环生成的，然后再去对这一部分程序进行编写。由于前面的正则表达式中数字和字母这两种情况直接各用一个单词替代，所以我们在生成词法分析程序的时候要将它们还原为（a|b|…|Y|Z）和（0|1|…|9）这种情况，这里面我们就另外用一个子函数去处理生成，

最后将编码输出文件规定好之后我们就完成了对该词法分析程序的代码的编写。编码方面我们还有一点需要注意的是，我们这个编码的源程序文件是要在任务二中进行解码后进行语法分析的，所以相应的编码识别我也规定了一个统一的标准。

该标准如下：

1. 遇到关键字，我们就原样输出，如“if:if”；
2. 遇到标识符，则保留原样输出，但前面加上字样“ID:”进行标识；
3. 数字则同理使用“NUMBER:”进行标识。
4. 遇到操作符，按照“名称：op”输出，如“LPAN:(”
5. 为了方便任务二，我们注释不进行输出。
6. 为了方便任务二，最后结尾加一个“EOF:EOF”

而我们生成的词法分析程序，采用是讲稿方法二来对状态进行处理，对应示例和伪代码如下：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

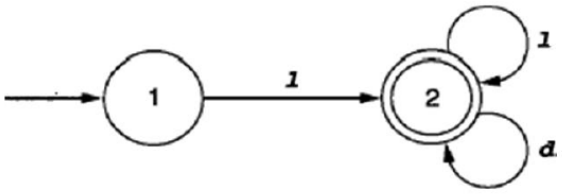
课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

输出：生成对应的词法分析程序(方法二)

```
state := 1; { start }
while state = 1 or 2 do
  case state of
  1:  case input character of
      letter : advance the input;
               state := 2;
      else state := . . . { error or other };
      end case;
  2:  case input character of
      letter, digit: advance the input;
                     state := 2; { actually unnecessary }
      else state := 3;
      end case;
  end case;
end while;
if state = 3 then accept else error ;
```



2.6.2 单元测试

华 南 师 范 大 学 实 验 报 告

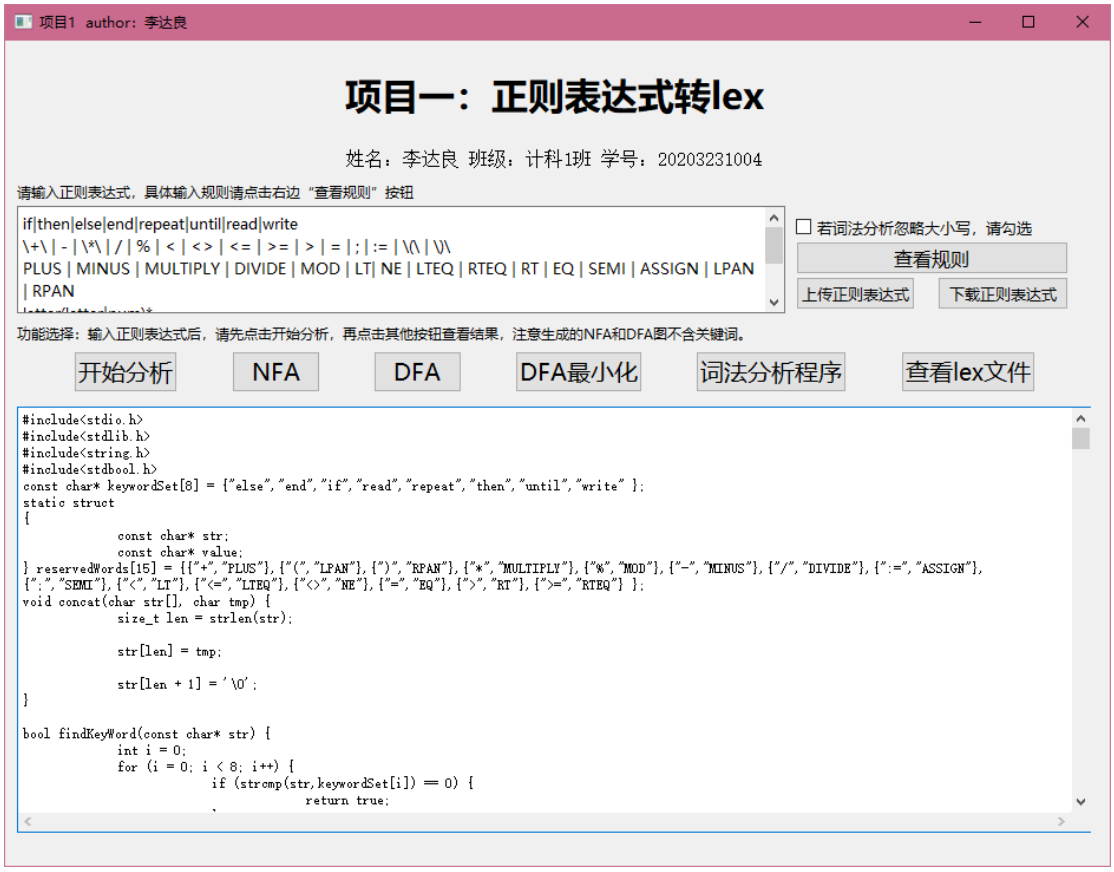
学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



得到代码后，我们对其进行编译，编译的方法：

可以直接使用 devC++/VS 等工具直接对其进行编译，注意得到的代码是 C 语言，不是 C++！

同时：sample.tny 必须和该程序放在同一个文件夹下。

编译后，跑出 lex 文件，进行查看：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.6.3 核心算法代码

```
bool genLexCase(QList<QString> tmpList, QString& codeStr, int idx, bool flag)
{
    bool rFlag = false;
    for (int i = 0; i < tmpList.size(); i++)
    {
        QString tmpKey = tmpList[i];
        char ch = tmpKey.toUtf8().constData()[0];
        if (dfaMinTable[idx].transitions[ch] == -1) {
            continue;
        }
        //字母情况
        if (tmpKey == QString((char)2))
        {
            for (int j = 0; j < 26; j++)
            {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
                codeStr += "\t\t\tcase \'" + QString(char('a' + j)) + "\':\n";
                codeStr += "\t\t\tcase \'" + QString(char('A' + j)) + "\':\n";
            }
            codeStr.chop(1);//去掉末尾字符
            if (flag)codeStr += "isIdentifier = true; ";
        }
        else if (tmpKey == QString((char)1))
        {
            //数字情况
            for (int j = 0; j < 10; j++)
            {
                codeStr += "\t\t\tcase \'" + QString::number(j) + "\':\n";
            }
            codeStr.chop(1);
            if (flag)codeStr += "isDigit = true; ";
        }
        else if (tmpKey == "~")
        {
            rFlag = true;
            continue;
        }
        else {
            QString keyRes = tmpKey;
            if (m1.find(ch) != m1.end()) {
                keyRes = QString::fromStdString(trim(m1[ch]));
            }
            codeStr += "\t\t\tcase \'" + keyRes + "\':";
        }

        if (flag) {
            codeStr += "state = " + QString::number(dfaMinTable[idx].transitions[ch]) +
"; ";
        }
        codeStr += "break;\n";
    }
    return rFlag;
}
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

3. 系统测试

见每个模块单元测试/测试报告

4. 系统问题分析

（1）正则表达式规则稍微有点复杂，可以考虑继续优化

5. 使用说明书

见使用说明书

（二）项目任务二：

1 系统设计

1.1 系统的关键数据结构

（1）文法解析

数据结构名称	grammarUnit	
数据结构类型	结构体	
数据结构用途	LR0 文法单元	
字段名称	字段类型	说明
gid	int	文法编号
left	string	文法->左边部分
right	string	文法->右边部分

数据结构名称	grammarDeque
数据结构类型	deque<grammarUnit>
数据结构用途	文法数组，用 deque 是因为可能得在头插入增广的文法

（2）求解 first 集合

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

数据结构名称	firstUnit	
数据结构类型	结构体	
数据结构用途	First 集合单元	
字段名称	字段类型	说明
s	set<string>	first 集合中的元素
isEpsilon	bool	是否包含空串

数据结构名称	firstSets
数据结构类型	map<string, firstUnit>
数据结构用途	First 集合结果

（3）Follow 集合

数据结构名称	followUnit	
数据结构类型	结构体	
数据结构用途	Follow 集合单元	
字段名称	字段类型	说明
s	set<string>	follow 集合中的元素

数据结构名称	followSets
数据结构类型	map<string, followUnit>
数据结构用途	Follow 集合结果

（4）LR0 DFA 表生成

数据结构名称	dfaCell
数据结构类型	结构体

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

数据结构用途	DFA 表每一项项目的结构	
字段名称	字段类型	说明
cellid	int	这一项的编号，便于后续判断状态相同
gid	int	文法编号
index	int	.在第几位，如 i=3, xxx.x，i=0, .xxxx, i=4, xxxx

数据结构名称	nextStateUnit	
数据结构类型	结构体	
数据结构用途	DFA 表下一项结构体	
字段名称	字段类型	说明
c	string	通过什么字符串进入这个状态
sid	int	下一个状态 id

数据结构名称	dfaState	
数据结构类型	结构体	
数据结构用途	DFA 表单个状态	
字段名称	字段类型	说明
sid	string	状态 id
originV	vector<int>	未闭包前的 cell
cellV	vector<int>	存储这个状态的 cellid
isEnd	bool	是否为规约状态
isSpecial	bool	是否是规约移进冲突

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

nextStateVector	vector<nextStateUnit>	下一个状态集合
right_VNs	set<string>	判断是否已经处理过这个非终结符

数据结构名称	dfaStateVector
数据结构类型	vector<dfaState>
数据结构用途	用于通过编号快速找到对应结构

（5）SLR1 分析

数据结构名称	SLRUnit	
数据结构类型	结构体	
数据结构用途	SLR1 表单元	
字段名称	字段类型	说明
m	map<string, string>	对应表格的每一项行列

数据结构名称	SLRVector
数据结构类型	vector<SLRUnit>
数据结构用途	SLR1 表行集合

（6）语法树生成

数据结构名称	BTreeNode	
数据结构类型	结构体	
数据结构用途	语法树结点	
字段名称	字段类型	说明
kind	string	结点类型

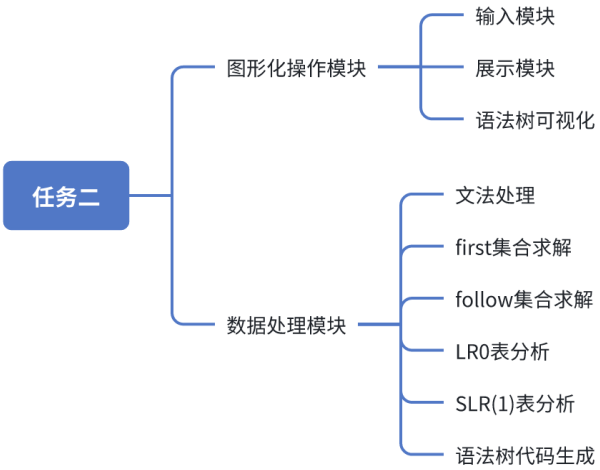
value	string	结点的值
nodeList	vector<BTreeNode*>	结点的孩子

1.2 系统功能结构

在进行项目设计时，我将项目任务二分成了两个大模块：图形化操作模块和数据处理模块。

其中图形化操作模块分为三个部分：输入模块、展示模块、语法树可视化模块；数据处理模块分为五部分：文法处理、First 集合求解、Follow 集合求解、LR0 表分析、SLR（1）表分析、语法树代码生成。

具体如图所示：



2 功能模块设计及实现

2.1 图形化操作模块

（1）描述

a. 输入模块：

通过 QT 实现 UI 的设计，左上角给予一个文法输入框，输入框上方有两个

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

按钮，分别是打开和保存，方便使用者的输入和保存文法。

b. 展示模块：

左下方是 First 集合和 Follow 集合求解，当输入了文法后，点击对应区域的求解按钮，就会显示文法中非终结符的 First 和 Follow 集合。中间部分是 LR(0)和 SLR(1)的求解区域，点击对应按钮，即可获得相应的答案，同时对于 LR(0)来说会有生成提示，来提示用户程序可能会对文法进行增广处理，对于 SLR(1)来说，当文法不符合 SLR(1)文法的时候，会提示不符合的原因。

c. 语法树可视化模块

使用者点击“代码生成”按钮，根据“生成提示”，准备相应的文件、选择对应的路径，生成语法树程序代码，通过编译运行语法树程序代码，得到语法树文件（tree.out），点击语法树展示即可展示语法树。

（2）QT 设计图

（3）界面截图

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.2 文法处理

2.2.1 描述

（1）我们规定文法输入格式如下：

* 第一行输入非终结符，用|隔开
* 第二行输入终结符，用|隔开
* 输入时不同的单词、标识符请用空格隔开；
* 用@表示空串，默认左边出现的第一个单词、标识符为文法的开始符号
* 同时，文法中含有或(|)，请分开两条输入

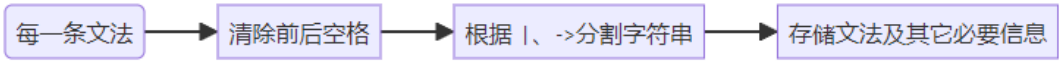
2. 进行分割处理：我会对非终结符和终结符进行分割，并保存。

3. 将文法结构化到 grammarMap 中，并为 LR0 自动机做准备。

4. 进行增广处理，如果开始符号有多个产生式，则增广文法。对文法进行编号，并构建 LR0 项目集规范族。

增广处理：如果开始符号的产生式多于一个，说明需要增广，使用 “zengguang” 作为增广后的非终结符。在 grammarDeque 中加入增广的产生式，并更新 LR0Result 字符串，提示用户程序进行了增广处理。

流程图如下：



2.2.2 测试示例

tiny 的文法，经过处理后，对应的输入文件如下所示：

```
program | stmt-sequence | statement | if-stmt | repeat-stmt | assign-stmt | read-stmt |
write-stmt | exp | simple-exp | comparison-op | addop | term | mulop | factor
SEMI | if | then | end | else | repeat | until | ASSIGN  | read | ID | number | write |
LT | EQ | LTEQ | NE | RTEQ | RT | PLUS | MINUS | MULTIPLY | DIVIDE | MOD
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
| LPAN | RPAN | NUMBER
program -> stmt-sequence
stmt-sequence -> stmt-sequence SEMI statement
stmt-sequence -> statement
statement -> if-stmt
statement -> repeat-stmt
statement -> assign-stmt
statement -> read-stmt
statement -> write-stmt
if-stmt -> if exp then stmt-sequence end
if-stmt -> if exp then stmt-sequence else stmt-sequence end
repeat-stmt -> repeat stmt-sequence until exp
assign-stmt -> ID ASSIGN exp
read-stmt -> read ID
write-stmt -> write exp
exp -> simple-exp comparison-op simple-exp
exp -> simple-exp
comparison-op -> LT
comparison-op -> EQ
comparison-op -> LTEQ
comparison-op -> NE
comparison-op -> RTEQ
comparison-op -> RT
simple-exp -> simple-exp addop term
simple-exp -> term
addop -> PLUS
addop -> MINUS
term -> term mulop factor
term -> factor
mulop -> MULTIPLY
mulop -> DIVIDE
mulop -> MOD
factor -> LPAN exp RPAN
factor -> NUMBER
factor -> ID
```

2.2.3 核心算法

```
void handleGrammar()
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
{
    vector<string> lines;
    istringstream iss(grammarStr);
    string line;
    // 防止中间有换行符
    while (getline(iss, line))
    {
        if (!line.empty())
        {
            lines.push_back(line);
        }
    }
    // 非终结符
    QString line1 = QString::fromStdString(lines[0]);
    QStringList tokens1 = line1.split("|");
    for (const QString& token : tokens1) {
        // 去除空格
        QString trimmedToken = token.trimmed();
        if (!trimmedToken.isEmpty()) {
            bigAlpha.push_back(trimmedToken.toStdString());
        }
    }
    // 终结符
    QString line2 = QString::fromStdString(lines[1]);
    QStringList tokens2 = line2.split("|");
    for (const QString& token : tokens2) {
        // 去除空格
        QString trimmedToken = token.trimmed();
        if (!trimmedToken.isEmpty()) {
            smallAlpha.push_back(trimmedToken.toStdString());
        }
    }
    for (int i = 2; i < lines.size(); i++)
    {
        string rule = lines[i];
        istringstream ruleStream(rule);
        string nonTerminal;
        ruleStream >> nonTerminal;  // 读取非终结符
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
        // 验证非终结符的格式
        if (!isBigAlpha(nonTerminal))
        {
            QMessageBox::critical(nullptr, "Error", "文法开头必须是非终结符!");
            continue;
        }
        // 跳过箭头符号 "->"
        string arrow;
        ruleStream >> arrow;
        string rightHandSide;
        getline(ruleStream, rightHandSide);
        // 去除开头的空格
        rightHandSide = rightHandSide.substr(1);
        // 如果是第一条规则，则认为是开始符号
        if (grammarMap.empty())
        {
            startSymbol = nonTerminal;
            trueStartSymbol = startSymbol;
        }
        // 将文法结构化

grammarMap[nonTerminal].insert(QString::fromStdString(rightHandSide).trimmed().toStdS
tring());
        // 为LR0做准备
        grammarDeque.push_back(grammarUnit(nonTerminal, rightHandSide));
    }
    // 增广处理
    if (grammarMap[startSymbol].size() > 1)
    {
        // 如果开始符号多于2个，说明需要增广，为了避免出现字母重复，采用`作为增广后的
字母，后期输出特殊处理
        grammarDeque.push_front(grammarUnit("zengguang", startSymbol));
        LR0Result += QString::fromStdString("进行了增广处理\n");
        trueStartSymbol = "zengguang";
    }
    // 开始编号
    int gid = 0;
    for (auto& g : grammarDeque)
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

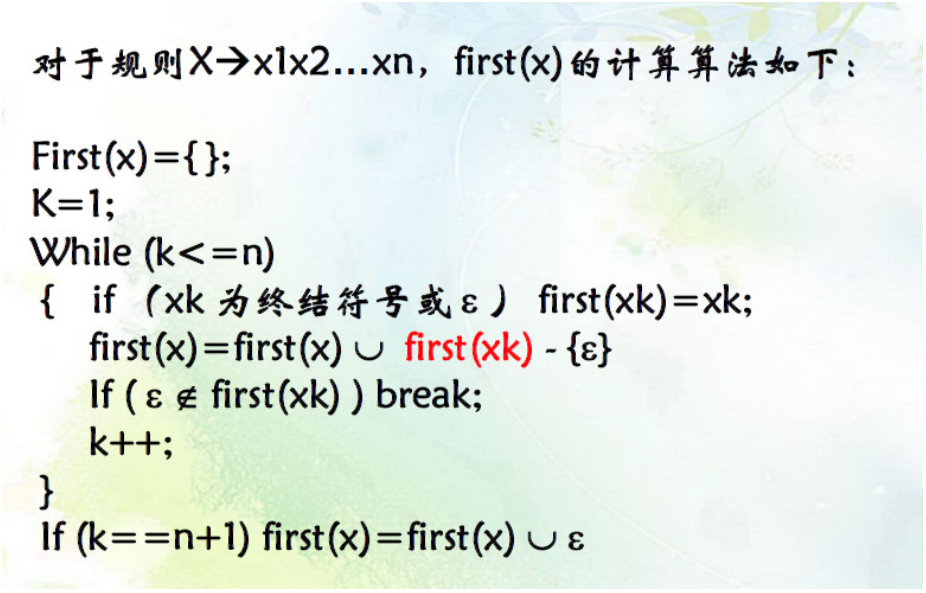
实验指导老师 黄煜廉 实验评分

```
    {
        g.gid = gid++;
        LR0Result += QString::number(g.gid) + QString::fromStdString(":")
            + QString::fromStdString(g.left == "zengguang" ? "E\'" : g.left) +
QString::fromStdString("->")
            + QString::fromStdString(g.right) + "\n";
        // 存入map中
        grammarToInt[make_pair(g.left, g.right)] = g.gid;
    }
}
```

2.3 First 集合求解

2.3.1 描述

对于 first 集合的求解，我们采用以下算法：



根据伪代码算法：

（1）我们可以先创建一个递归算法：遍历文法中的每个非终结符，对于每个非终结符，遍历其产生式。对于每个产生式，找到其每个符号的 First 集合，并将其加入到该非终结符的 First 集合中。如果发现某个符号的 First 集合包含空串，并且该符号后还有其他符号，则继续处理下一个符号。如果某个产生式的所有符号的 First 集合都包含空串，则将空串加入该非终结符的 First 集合中。检查原始 First 集合的大小和是否包含空串，如果发生变化，则将 flag 置为 true。

流程图如下：

华 南 师 范 大 学 实 验 报 告

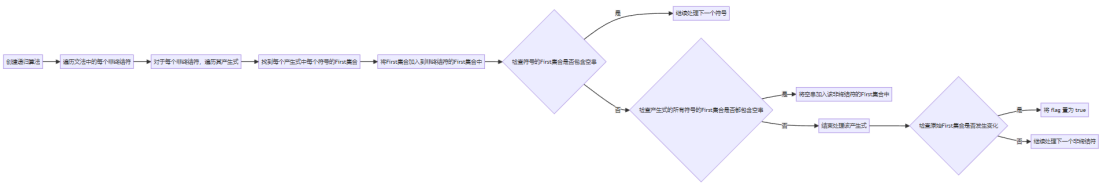
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

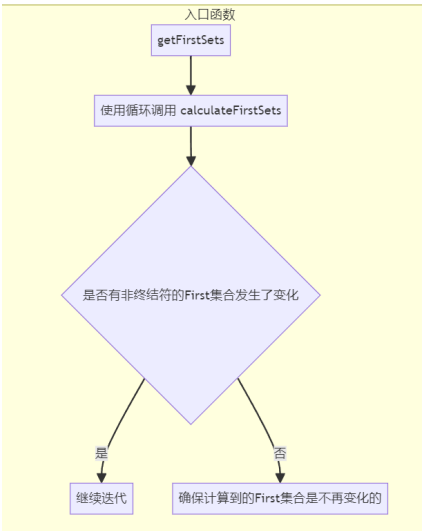
实验指导老师 黄煜廉 实验评分



（2）同时我们给一个入口，getFirstSets() 函数：

使用循环调用 calculateFirstSets() 直到不再有变化。在每次迭代中，检查是否有非终结符的 First 集合发生了变化，如果有变化则继续迭代。通过这种方式，确保计算到的 First 集合是不再变化的。

流程图如下：



2.3.2 单元测试

	非终结符	First集合
1	addop	MINUS,PLUS
2	assign-stmt	ID
3	comparison-op	EQ,LT,LTEQ,N...
4	exp	ID,LPAN,NUM...

	非终结符	First集合
12	statement	ID,if,read,rep...
13	stmt-sequence	ID,if,read,rep...
14	term	ID,LPAN,NUM...
15	write-stmt	write

数据较多，在此不一一展示。

2.3.3 核心算法代码

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
bool calculateFirstSets()
{
    bool flag = false;
    for (auto& grammar : grammarMap)
    {
        string nonTerminal = grammar.first;
        // 保存当前First集合的大小，用于检查是否有变化
        size_t originalSize = firstSets[nonTerminal].s.size();
        bool originalE = firstSets[nonTerminal].isEpsilon;
        for (auto& g : grammar.second)
        {
            QStringList gList = QString::fromStdString(g).split(" ");
            int k = 0;
            while (k <= gList.size() - 1)
            {
                string t = gList[k].toStdString();
                set<string> first_k;
                if (t == "@")
                {
                    k++;
                    continue;
                }
                else if (isSmallAlpha(t))
                {
                    first_k.insert(t);
                }
                else
                {
                    first_k = firstSets[t].s;
                }
                firstSets[nonTerminal].s.insert(first_k.begin(), first_k.end());
                // 如果是终结符或者没有空串在非终结符中，直接跳出
                if (isSmallAlpha(t) || !firstSets[t].isEpsilon)
                {
                    break;
                }
                k++;
            }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
            if (k == g.size())
            {
                firstSets[nonTerminal].isEpsilon = true;
            }
        }
        // 看原始大小和是否变化epsilon，如果变化说明还得重新再来一次
        if (originalSize != firstSets[nonTerminal].s.size() || originalE !=
firstSets[nonTerminal].isEpsilon)
        {
            flag = true;
        }
    }
    return flag;
}

void getFirstSets()
{
    // 不停迭代，直到First集合不再变化
    bool flag = false;
    do
    {
        flag = calculateFirstSets();
    } while (flag);
}
```

2.4 follow 集合求解

2.4.1 描述

针对 Follow 集合求解，我们采用以下算法：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

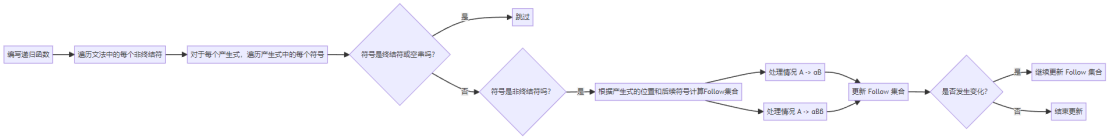
课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

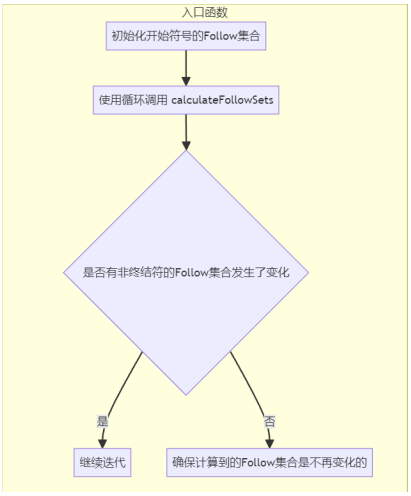
（1）我们编写一个递归函数，遍历文法中的每个非终结符，对于每个产生式，遍历产生式中的每个符号。如果符号是终结符或空串，则跳过。如果符号是非终结符，根据产生式的位置和后续符号计算 Follow 集合。通过两个情况（A -> α B 和 A -> α B β）来处理 Follow 集合的计算。更新 Follow 集合，并检查是否发生变化。

流程图如下：



（2）编写一个入口，初始化开始符号的 Follow 集合为 { '$' }。使用循环调用 calculateFollowSets() 直到不再有变化。在每次迭代中，检查是否有非终结符的 Follow 集合发生了变化，如果有变化则继续迭代。通过这种方式，确保计算到的 Follow 集合是不再变化的。

流程图如下：



华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.4.2 单元测试

	非终结符	Follow集合
1	addop	ID,LPAN,NUM..
2	assign-stmt	$,SEMI,else,e...
3	comparison-op	ID,LPAN,NUM..
4	exp	$,RPAN,SEMI,..

	非终结符	Follow集合
12	statement	$,SEMI,else,e...
13	stmt-sequence	$,SEMI,else,e...
14	term	$,DIVIDE,EQ,L..
15	write-stmt	$,SEMI,else,e...

数据较多，在此不一一展示。

2.4.3 核心算法代码

```
bool calculateFollowSets()
{
    bool flag = false;
    for (auto& grammar : grammarMap)
    {
        string nonTerminal = grammar.first;
        for (auto& g : grammar.second)
        {
            QStringList gList = QString::fromStdString(g).split(" ");
            for (int i = 0; i < gList.size(); ++i)
            {
                string t = gList[i].toStdString();
                if (isSmallAlpha(t) || t == "@")
                {
                    continue;  // 跳过终结符
                }
                set<string> follow_k;
                size_t originalSize = followSets[t].s.size();
                if (i == gList.size() - 1)
                {
                    // Case A: A -> αB, add Follow(A) to Follow(B)
                    follow_k.insert(followSets[nonTerminal].s.begin(),
followSets[nonTerminal].s.end());
                }
                else
                {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
                    // Case B: A -> αBβ
                    int j = i + 1;
                    while (j < gList.size())
                    {
                        string t2 = gList[j].toStdString();
                        if (isSmallAlpha(t2))
                        {   // 终结符直接加入并跳出
                            follow_k.insert(t2);
                            break;
                        }
                        else
                        {   // 非终结符加入first集合
                            set<string> first_beta = firstSets[t2].s;
                            follow_k.insert(first_beta.begin(), first_beta.end());
                            // 如果没有空串在first集合中，停止。
                            if (!firstSets[t2].isEpsilon)
                            {
                                break;
                            }
                            ++j;
                        }
                    }
                    // If β is ε or β is all nullable, add Follow(A) to Follow(B)
                    if (j == gList.size())
                    {
                        follow_k.insert(followSets[nonTerminal].s.begin(),
followSets[nonTerminal].s.end());
                    }
                }
                addToFollow(t, follow_k);
                // 检查是否变化
                if (originalSize != followSets[t].s.size())
                {
                    flag = true;
                }
            }
        }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
    }
    return flag;
}
```

2.5 LR0 表分析

2.5.1 描述

（1）对于 LR(0)生成，我们首先先对第一条文法进行生成 LR0 状态，并且因为增广文法的存在，一定只会有一个入口，即一条文法。

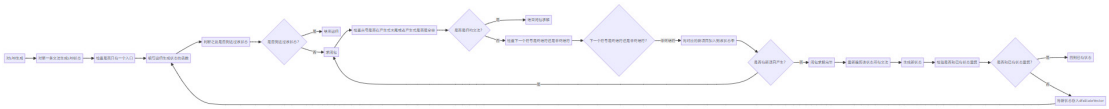
（2）得到第一个状态后，我们可以编写递归生成状态的函数了，主要步骤类似于 DFS 遍历，首先判断我们之前是否到达过该状态，如果到达过，说明我们不需要再对该状态递归下去，防止死循环。

（3）然后，我们要对当前走到的状态求闭包：我们要检查点号是否在产生式末尾或者产生式是否是空串，如果是的话，说明是归约文法，不是的话，检查下一个符号是终结符还是非终结符，非终结符的话，我们需要将对应的新项目加入到一个该状态中，直到没有新项目产生，这时闭包求解完毕。

（4）求解完毕之后，重新遍历该状态所有文法，生成新状态，即每个文法往前走一步，但不能直接存入新状态的 dfaStateVector，要检验是否和 dfaStateVector 中某个状态是一样的，如果是一样的话，本状态在该项目中往前走一步应该是回到 dfaStateVector 中某个状态上。

（5）最后我们对下一个状态进行递归 DFS。

流程图如下：



2.5.2 单元测试

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.5.3 核心算法代码

```
// 生成LR0状态
void generateLR0State(int stateId)
{
    // DFS,如果走过就不走了
    if (visitedStates.count(stateId) > 0) {
        return;
    }
    // 标记走过了
    visitedStates.insert(stateId);
    // 求闭包
    for (int i = 0; i < dfaStateVector[stateId].cellV.size(); ++i)
    {
        dfaCell& currentCell = dfaCellVector[dfaStateVector[stateId].cellV[i]];

        qDebug() << QString::fromStdString(grammarDeque[currentCell.gid].left) <<
QString::fromStdString("->") <<
QString::fromStdString(grammarDeque[currentCell.gid].right) << endl;

        qDebug() << "current index:" << currentCell.index << endl;

        QStringList rightList =
QString::fromStdString(grammarDeque[currentCell.gid].right).split(" ");

        // 如果点号在产生式末尾或者空串，则跳过（LR0不需要结束）
        if (currentCell.index == rightList.size() ||
grammarDeque[currentCell.gid].right == "@")
        {
            dfaStateVector[stateId].isEnd = true;
            continue;
        }
        string nextSymbol = rightList[currentCell.index].toStdString();
        // 如果nextSymbol是非终结符，则将新项添加到状态中
        if (isBigAlpha(nextSymbol) &&
dfaStateVector[stateId].right_VNs.find(nextSymbol) ==
dfaStateVector[stateId].right_VNs.end())
        {
            dfaStateVector[stateId].right_VNs.insert(nextSymbol);
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
            for (auto& grammar : grammarMap[nextSymbol])
            {
                // 获取通过nextSymbol转移的新LR0项
                dfaCell nextCell = dfaCell();
                nextCell.gid = grammarToInt[make_pair(nextSymbol, grammar)];
                nextCell.index = 0;
                int nextcellid = isNewCell(nextCell.gid, nextCell.index);
                if (nextcellid == -1)
                {
                    nextCell.cellid = ccnt++;
                    dfaCellVector.push_back(nextCell);
                    dfaStateVector[stateId].cellV.push_back(nextCell.cellid);
                }
                else dfaStateVector[stateId].cellV.push_back(nextcellid);
            }

        }
    }
    // 暂存新状态
    map<string, dfaState> tempSave;
    // 生成新状态，但还不能直接存到dfaStateVector中，我们要校验他是否和之前的状态一样
    for (int i = 0; i < dfaStateVector[stateId].cellV.size(); ++i)
    {
        dfaCell& currentCell = dfaCellVector[dfaStateVector[stateId].cellV[i]];
        QStringList rightList =
QString::fromStdString(grammarDeque[currentCell.gid].right).split(" ");
        // 如果点号在产生式末尾，则跳过（LR0不需要结束）
        if (currentCell.index == rightList.size() ||
grammarDeque[currentCell.gid].right == "@")
        {
            continue;
        }
        // 下一个字符
        string nextSymbol = rightList[currentCell.index].toStdString();
        // 创建下一个状态（临时的）
        dfaState& nextState = tempSave[nextSymbol];
        dfaCell nextStateCell = dfaCell();
        nextStateCell.gid = currentCell.gid;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
        nextStateCell.index = currentCell.index + 1;
        // 看看里面的项目是否有重复的，如果重复拿之前的就好，不重复生成
        int nextStateCellid = isNewCell(nextStateCell.gid, nextStateCell.index);
        if (nextStateCellid == -1)
        {
            nextStateCell.cellid = ccnt++;
            dfaCellVector.push_back(nextStateCell);
        }
        else nextStateCell.cellid = nextStateCellid;
        nextState.cellV.push_back(nextStateCell.cellid);
        nextState.originV.push_back(nextStateCell.cellid);
        // 收集一下，方便后面画表
        if (isBigAlpha(nextSymbol))
        {
            VN.insert(nextSymbol);
        }
        else if (isSmallAlpha(nextSymbol))
        {
            VT.insert(nextSymbol);
        }
    }
    // 校验状态是否有重复的
    for (auto& t : tempSave)
    {
        dfaState nextState = dfaState();
        int newStateId = isNewState(t.second.originV);
        // 不重复就新开一个状态
        if (newStateId == -1)
        {
            nextState.sid = scnt++;
            nextState.cellV = t.second.cellV;
            nextState.originV = t.second.originV;
            dfaStateVector.push_back(nextState);
        }
        else nextState.sid = newStateId;
        // 存入现在这个状态的nextStateVector
        nextStateUnit n = nextStateUnit();
        n.sid = nextState.sid;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
        n.c = t.first;
        dfaStateVector[stateId].nextStateVector.push_back(n);
    }
    // 对每个下一个状态进行递归
    int nsize = dfaStateVector[stateId].nextStateVector.size();
    for (int i = 0; i < nsize; i++)
    {
        auto& nextunit = dfaStateVector[stateId].nextStateVector[i];
        generateLR0State(nextunit.sid);
    }
}
```

2.6 SLR1 状态表生成

2.6.1 描述

（1）首先我们要检验文法是否属于 SLR(1)文法，主要判断规则如下：

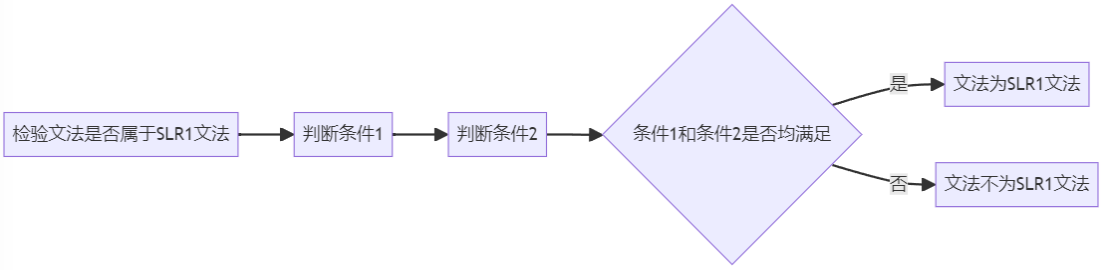
•当且仅当对于任何状态 s，以下的两个条件：

•1) 对于在 s 中的任何项目 A→a.Xb，当 X 是一个终结符，且 X 在 Follow (B) 中时，s 中没有完整的项目 B→c.。 [移进-归约冲突]

•2) 对于在 s 中的任何两个完整项目 A→a.和 B→b.，Follow(A)∩Follow(B)为空。[归约-归约冲突]

均满足时，文法为 SLR(1)文法。

流程图如下：



（2） 最后我们构建 SLR(1)分析表，其实 SLR(1)分析表是在 LR(0)基础上构建得来的，因此，只是对 LR(0)内数据进行一些处理和特殊判断，如归约状态的处理、最终接受的处理，同时还需要求解 FOLLOW 集合，来使归约状态得到更好的判断。

流程图如下：

华 南 师 范 大 学 实 验 报 告

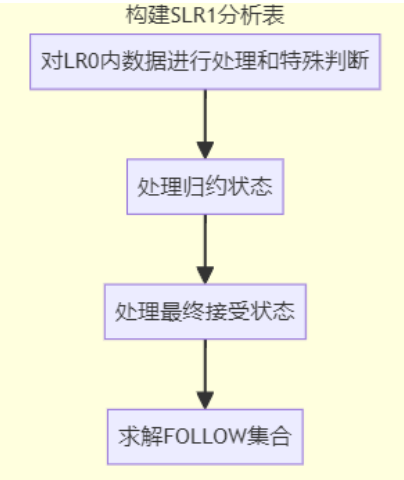
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



（3）对于移进归约冲突，我们默认选择只做移进不做规约

2.6.2 单元测试

	状态	$	ASSIGN	DIVIDE	EQ	ID	LPAN	LT
1	0					s1		
2	1		s13					
3	2	r(statement-...						
4	3					s14	s15	
5	4	r(statement->...						
6	5					s43		
7	6	r(statement-...						

2.6.3 核心算法代码

```
int getSLR1Table()
{
    // SLR1分析错误就直接停止
    int r = SLR1Analyse();
    if (r != 0 && r != 1) return r;
    // 如果分析正确，通过LR0构造SLR1分析表（必须先调用getLR0）
    for (const dfaState& ds : dfaStateVector)
    {
        SLRUnit slrunit = SLRUnit();
        // 如果是归约，得做特殊处理
        if (ds.isEnd)
        {
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
            if (!ds.isSpecial) // 对于移进规约冲突的项，只做移进不做规约
            {
                // 规约项目的非终结符
                string gl;
                string gr;
                // 规约状态
                for (int cellid : ds.cellV)
                {
                    // 拿到这个cell
                    const dfaCell& cell = dfaCellVector[cellid];
                    // 获取文法
                    const grammarUnit gm = grammarDeque[cell.gid];

                    QStringList rightList =
QString::fromStdString(grammarDeque[cell.gid].right).split(" ");

                    // 判断是不是规约项目
                    if (cell.index == rightList.size() || gm.right == "@")
                    {
                        gl = gm.left;
                        gr = gm.right;
                        break;  // 前面的SLR1校验保证了只有一个归约项目
                    }
                }
                // 得到这个非终结符Follow集合
                set<string> follow = followSets[gl].s;
                // follow集合每个元素都能归约
                for (string ch : follow)
                {
                    if (gl == trueStartSymbol) slrunit.m[ch] = "ACCEPT";
                    else slrunit.m[ch] = "r(" + gl + "->" + gr + ")";
                }
            }
            // 对于下一个节点（可能会存在）
            for (const auto& next : ds.nextStateVector)
            {
                string ch = next.c;
                int sid = next.sid; //  下一个状态id
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
                // 获取下一个状态具体信息
                dfaState d = dfaStateVector[sid];
                if (isBigAlpha(ch))
                {
                    slrunit.m[ch] = to_string(sid);
                }
                else
                {
                    slrunit.m[ch] = "s" + to_string(sid);
                }
            }
        }
        else
        {
            for (const auto& next : ds.nextStateVector)
            {
                string ch = next.c;
                int sid = next.sid; //  下一个状态id
                // 获取下一个状态具体信息
                dfaState d = dfaStateVector[sid];
                if (isBigAlpha(ch))
                {
                    slrunit.m[ch] = to_string(sid);
                }
                else
                {
                    slrunit.m[ch] = "s" + to_string(sid);
                }
            }
        }
        SLRVector.push_back(slrunit);
    }
    return r;
}
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.7 语法树生成

2.7.1 描述

对于语法树生成，我们采用和任务一相似的思路，生成一个语法树生成程序的代码，然后编译运行后得到 tree.out 的语法树文本文件，通过语法树可视化模块，对这个文本文件进行一个树形结构的可视化。具体流程如下：

（1） 生成一个语法树生成程序的代码，首先先把 SLR1 分析的结果转移到生成程序的代码中，否则无法根据 SLR1 表来进行移进和规约处理。这里面，我采取的是类似于 json 序列化和反序列化的方法，将 SLR1 表以特定格式进行输出，然后在语法树程序里面解析特定格式，使得程序能根据 SLR1 结构来进行移进和规约。如下图就是一个示例：

```
SLRUnit
{
    Key: ID
    Value: s1
    Key: assign-stmt
    Value: 2
    Key: if
    Value: s3
    Key: if-stmt
    Value: 4
    Key: read
    Value: s5
    Key: read-stmt
    Value: 6
    Key: repeat
    Value: s7
    Key: repeat-stmt
    Value: 8
    Key: statement
    Value: 9
    Key: stmt-sequence
    Value: 10
    Key: write
    Value: s11
    Key: write-stmt
    Value: 12
}
```

但这样子会存在一个问题，SLR1 输出的格式太长，C++编译器在编译阶段对字符串的初始化长度有限制，于是我采用输出到 txt 文件，然后读入文件流的方式来解决，通过生成 SLR1Str.txt，并在语法树程序中进行解析。

（2）当语法树程序中，可以通过 SLR1 表进行移进归约操作的时候，我们采取分析栈的操作来解析整个程序。我们可以把整段程序当作一个大号的字符串，通过分析栈来进行分析。具体而言，分析栈执行流程如下：

a. 提取 lex 文件中每一行的 key 和 value 值。

b. 获取当前状态栈的顶部状态。根据当前状态和输入的键值（如果是 EOF，则转换为 $）在状态转换表 SLRVector 中查找下一个状态。

c. 根据下一个状态的类型进行不同的处理：

- 如果是 's'（shift），将输入的字符压入字符栈 strStack，并将下一个状态压入状态栈 stateStack。
- 如果是 'r'（reduce），解析规约的动作，并调用相应的函数。然后根

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

据规约后的符号和当前状态在状态转换表中查找下一个状态，并将其压入状态栈，并将规约后的符号压入字符栈。

- 如果是 'A'（accept），输出成功信息。
- 如果是其他情况，则输出状态表出错信息

（3）在解析规约的动作，调用对应函数时，此时就应该生成语法树对应的结构。我们在生成语法树程序代码前，需要先输入语法树对应的语义函数。我们规定如下：

a. 每一行都与文法一一对应，即第一行输入第一行文法的语义函数，第二行输入第二行文法语义函数，以此类推。

b. 针对每一行文法的“->”右边部分，每个标识符对应-1 0 1 其中一个，具体含义如下：

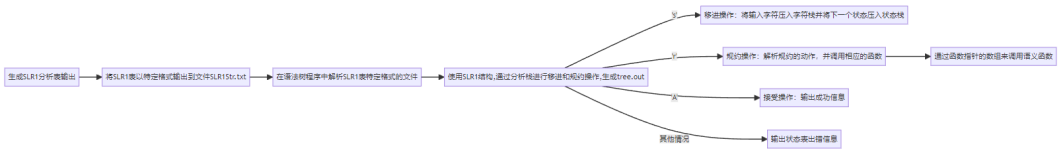
- -1 代表不生成这个节点
- 0 代表是根
- 1+ 代表第几个孩子，孩子的顺序

c. 如果文法存在左递归，程序自动默认规约的字符的根节点。

（4）调用对应函数，我们采取的方法是，定义一个存储函数指针的数组 string (*funcArray[])()，使用了一个 grammarMap 来存放文法规约时对应调用函数的数组下标，使得解析规约动作的时候，可以快速调用对应函数，完成语义动作的执行。

（5）最后，生成了 tree.out 代码，同样使用一个类似与 json 格式的输出，再使用 QT 进行读入可视化。

整体流程图如下图所示：



2.7.2 单元测试

（1）语义函数示例：

华 南 师 范 大 学 实 验 报 告

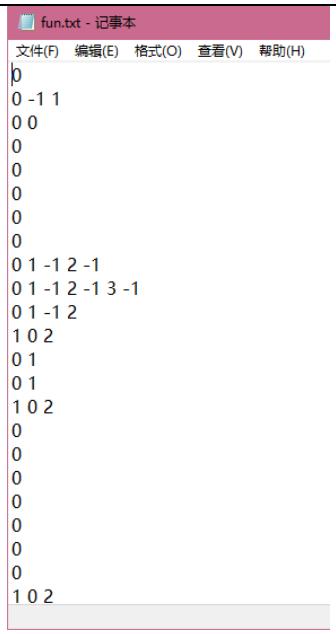
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

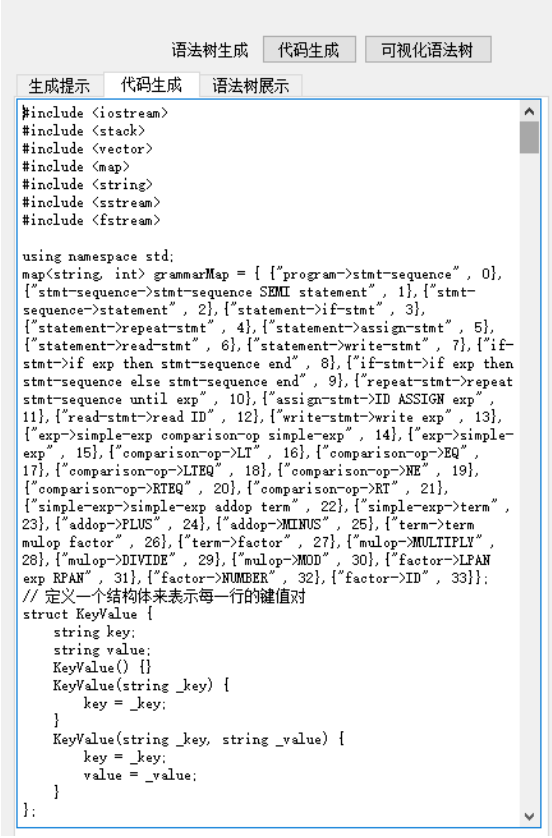
课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



（2）生成代码示例



华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

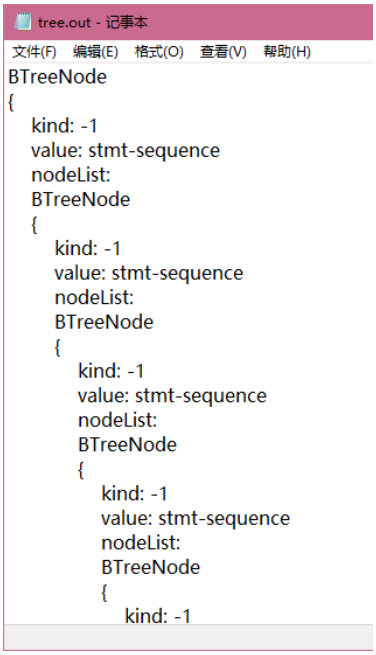
专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

（3）生成语法树 tree.out 示例：



（4）语法树可视化示例：

华 南 师 范 大 学 实 验 报 告

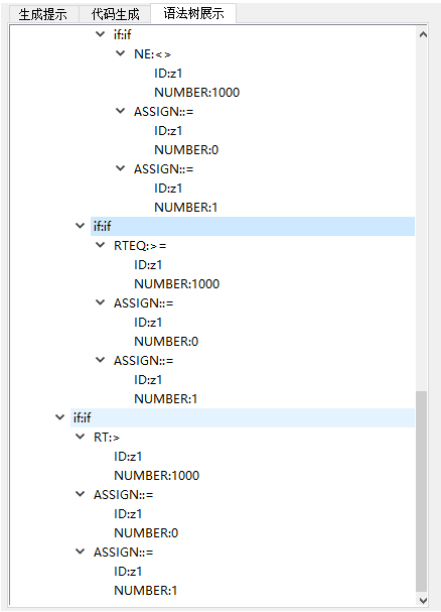
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



2.7.3 核心代码

```
QString generateFunCode(QString funQStr) {
    QString funCode;

    // 先进行分行处理
    vector<string> lines;
    istringstream iss(funQStr.toStdString());
    string line;

    if (grammarDeque[0].left == "zengguang") {
        lines.push_back("zengguang");
    }

    // 防止中间有换行符
    while (getline(iss, line))
    {
        if (!line.empty())
        {
            lines.push_back(line);
        }
    }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
    if (lines.size() != grammarDeque.size()) {
        funCodeError = 1;   // 如果行数不一致代表有问题
        return "";
    }

    for (int i = 0; i < lines.size(); i++) {

        map<int, int> treeNodeMap;
        int count = 0;
        if (lines[i] == "zengguang") continue;

        QStringList numList = QString::fromStdString(lines[i]).split(" ");
        QStringList stringList = QString::fromStdString(grammarDeque[i].right).split("
");

        for (int j = 0; j < stringList.size(); j++) {
            int index = stoi(numList[j].toStdString());
            if (index == -1) continue;
            else {
                treeNodeMap[index] = j;
                count++;
            }
        }

        funCode += "// ";
        funCode += QString::fromStdString(grammarDeque[i].left + "->" +
grammarDeque[i].right);
        funCode += "\n";
        funCode += "string fun";
        funCode += QString::number(i);
        funNumber.push_back(i);
        funCode += "() {\n";

        for (int j = stringList.size() - 1; j >= 0; j--) {
            if (isBigAlpha(stringList[j].toStdString())) {  // 非终结符
                funCode += "\tBTreeNode* newNode";
                funCode += QString::number(j);
                funCode += R"( = treeStack.top();
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
strStack.pop();
stateStack.pop();
treeStack.pop();)";
            funCode += "\n";
        }
        else if (isSmallAlpha(stringList[j].toStdString())) {   // 终结符
            funCode += "\tstring symbol";
            funCode += QString::number(j);
            funCode += R"( = strStack.top().value;
strStack.pop();
stateStack.pop();)";
            funCode += "\n";
            funCode += "\tBTreeNode* newNode";
            funCode += QString::number(j);
            funCode += " = new BTreeNode(\"";
            funCode += QString::fromStdString(stringList[j].toStdString());
            funCode += "\", symbol";
            funCode += QString::number(j);
            funCode += ");\n";
        }
        funCode += "\n";


    }

    // 左递归
    if (grammarDeque[i].left == stringList[0].toStdString()) {
        funCode += "\tBTreeNode* rootNode = new BTreeNode(\"-1\",\"";
        funCode += QString::fromStdString(grammarDeque[i].left);
        funCode += "\");\n ";
        for (int k = 0; k < count; k++) {
            funCode += "\trootNode";
            funCode += "->nodeList.push_back(newNode";
            funCode += QString::number(treeNodeMap[k]);
            funCode += ");\n";
        }
        funCode += "\ttreeStack.push(rootNode);\n";
    }
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
        else {
            for (int k = 1; k < count; k++) {
                funCode += "\tnewNode";
                funCode += QString::number(treeNodeMap[0]);
                funCode += "->nodeList.push_back(newNode";
                funCode += QString::number(treeNodeMap[k]);
                funCode += ");\n";
            }

            if (grammarDeque[i].right != "@") {
                funCode += "\ttreeStack.push(newNode";
                funCode += QString::number(treeNodeMap[0]);
                funCode += ");\n";
            }
            else
            {
                funCode += "\tBTreeNode* rootNode = new BTreeNode(\"-2\",\"";
                funCode += QString::fromStdString(grammarDeque[i].left);
                funCode += "\");\n ";
                funCode += "\ttreeStack.push(rootNode);\n";
            }
        }



        funCode += "\treturn \"";
        funCode += QString::fromStdString(grammarDeque[i].left);
        funCode += "\";\n";

        funCode += "}\n\n";

    }

    return funCode;
}
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

3 系统测试

见每个模块单元测试/测试报告

4. 系统问题分析

（1）采用代码生成的方式，稍微有点繁琐，可以考虑将代码生成变成内嵌至 QT 中。

5. 使用说明书

见使用说明书

（三）项目任务三：

1. 项目任务一测试

1.1 准备工作

minic 的正则表达式：

```
else|if|int|float|double|return|void|do|while
\+\ | - | \*\ | / | % | <  | <= | >= | > | ==| != | = | ; | , | \(\ | \)\ | \[\ | \]\ | { | }
PLUS | MINUS | MULTIPLY | DIVIDE | MOD | LT| LTEQ | RTEQ | RT | EQ | NE | ASSIGN | SEMI | DOU | LLM | RLM | LMM | RMM | LBM | RBM
(_|letter)(_|letter|num)*
num num* (.num num *)?
//~\n
```

sample.tny 文件：

```
//Sample program
//In MiniC language-computes factorial
//}
int x; // an integer
int fact;
double y;
int arr[100];
void test(int a, int b)
{
  a = 1;
  b = 2;
  return ;
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
}
void main(void)
{
  x = 2;
  y = 3;
  if ( x > 1 )  //don't compute if x <= 0
      do
          fact = fact * x;
      while (fact >= 1);
  else
    x = 1;
  return ; //return void
}
```

输入正则表达式：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

1.2 测试结果：NFA

	标志	ID	num	letter	+	(	)	*	[	]	\n	!	#	%	,	-	.	/	;	<	=	>	_	{	}	非\n
1	-	134											122,124													
2		122											104,106													
3		124																125								
4		125											126													
5		126																127								
6		127											130													
7		130											128,131													
8		128																								129
9		131											132													
10		132									133															
11		133											135													

1.3 测试结果：DFA 图

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

1.4 测试结果：DFA 图最小化

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

	标志	ID	num	letter	+	(	)	*	[	]	\n	!	%	,	-	.	/	;	<	=	>	_	{	}	非\n
1	-	0	1	5	4	4	4	4	4	4		2	4	4	4		8	4	9	9	9	5	4	4	
2	+	1	1													3									
3		2																		4					
4		3	6																						
5	+	4																							
6	+	5	5	5																		5			
7	+	6	6																						
8		7									4														7
9	+	8															7								
10	+	9																		4					

1.5 测试结果：生成词法程序

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

代码太长，具体可查看：

2. 测试文件夹\1. 任务一测试文本及程序\2. 程序生成文件中的_lexer.c 文件

1.6 编译_lexer.c 文件并运行

注意：sample.tny 必须和该程序放在同一个文件夹下。

华 南 师 范 大 学 实 验 报 告

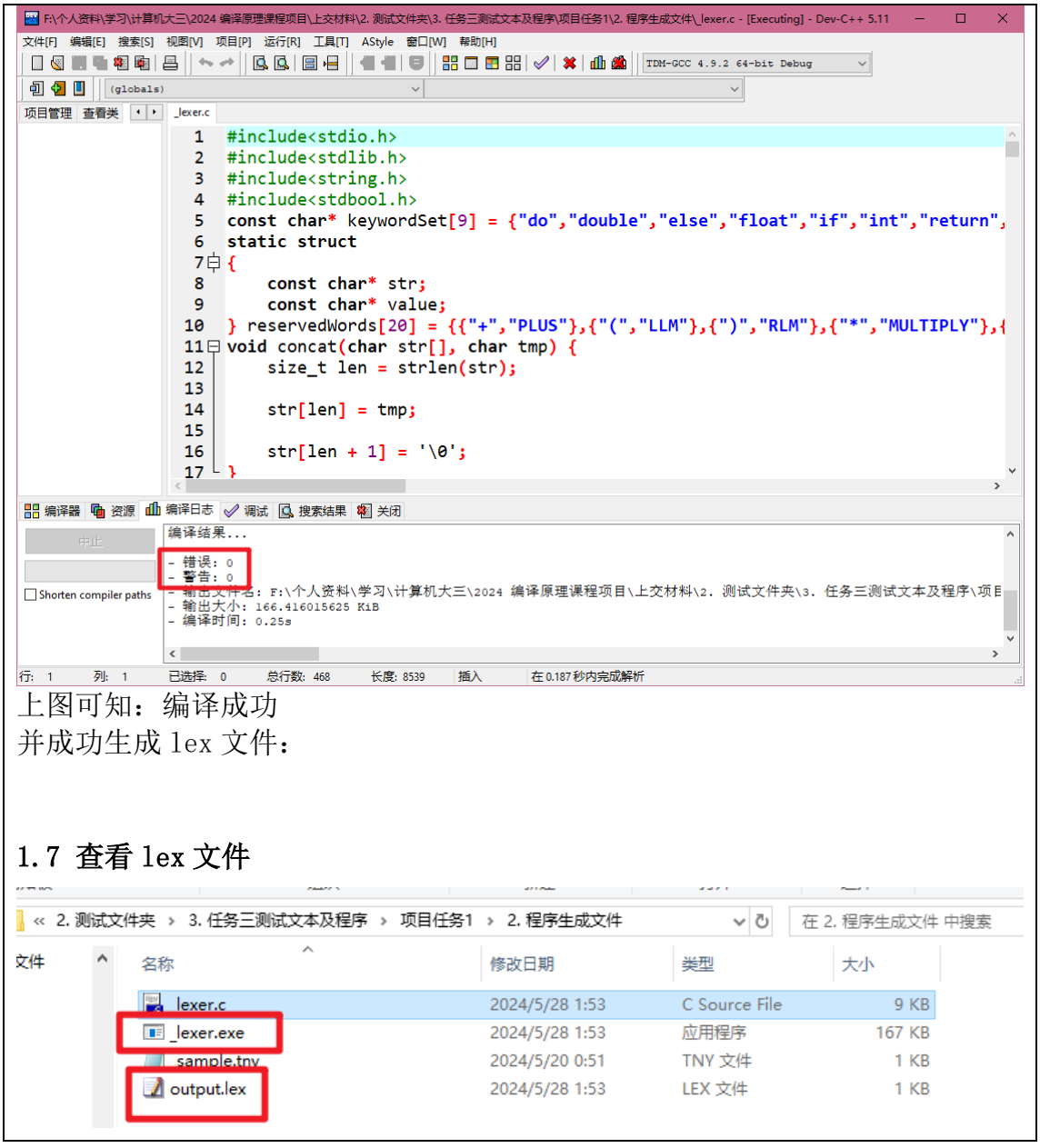
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



上图可知：编译成功

并成功生成 lex 文件：

1.7 查看 lex 文件

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

具体 lex 如下：

```
int:int
ID:x
SEMI:;
int:int
ID:fact
SEMI:;
double:double
ID:y
SEMI:;
int:int
ID:arr
LMM:[
NUMBER:100
RMM:]
SEMI:;
void:void
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 **20203231004**

专　业 计算机科学与技术 年级、班级 **2021 级 1 班**

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 **2024** 年 **5** 月 **25** 日

实验指导老师 黄煜廉 实验评分

```
ID:test
LLM:(
int:int
ID:a
DOU:,
int:int
ID:b
RLM:)
LBM:{
ID:a
ASSIGN:=
NUMBER:1
SEMI:;
ID:b
ASSIGN:=
NUMBER:2
SEMI:;
return:return
SEMI:;
RBM:}
void:void
ID:main
LLM:(
void:void
RLM:)
LBM:{
ID:x
ASSIGN:=
NUMBER:2
SEMI:;
ID:y
ASSIGN:=
NUMBER:3
SEMI:;
if:if
LLM:(
ID:x
RT:>
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
NUMBER:1
RLM:)
do:do
ID:fact
ASSIGN:=
ID:fact
MULTIPLY:*
ID:x
SEMI:;
while:while
LLM:(
ID:fact
RTEQ:>=
NUMBER:1
RLM:)
SEMI:;
else:else
ID:x
ASSIGN:=
NUMBER:1
SEMI:;
return:return
SEMI:;
RBM:}
EOF:EOF
```

对照 minic 源程序，可知解析完全正确。

测试结果

项目任务三-项目一的测试完全通过

2. 项目任务二的测试

2.1 准备工作

minic 的文法：

```
program | definition-list | definition | variable-definition |
function-definition | type-indicator | parameters | compound-stmt |
parameter-list | parameter | local-definitions | statement-list |
statement | expression-stmt | condition-stmt | dowhile-stmt |
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

return-stmt | expression | simple-expression | variable | additive-expression | relop | term | addop | mulop | factor | call | arguments | argument-list

ID | SEMI | LMM | RMM | int | float | double | void | LLM | RLM | if | else | do | while | return | LTEQ | LT | RT | RTEQ | EQ | NE | PLUS | MINUS | MULTIPLY | DIVIDE | MOD | NUMBER | DOU | LBM | RBM | ASSIGN

program -> definition-list

definition-list -> definition-list definition

definition-list -> definition

definition -> variable-definition

definition -> function-definition

variable-definition -> type-indicator ID SEMI

variable-definition -> type-indicator ID LMM NUMBER RMM SEMI

type-indicator -> int

type-indicator -> float

type-indicator -> double

type-indicator -> void

function-definition -> type-indicator ID LLM parameters RLM compound-stmt

parameters -> parameter-list

parameters -> void

parameter-list -> parameter-list DOU parameter

parameter-list -> parameter

parameter -> type-indicator ID

parameter -> type-indicator ID LMM RMM

compound-stmt -> LBM local-definitions statement-list RBM

local-definitions -> local-definitions variable-definition

local-definitions -> @

statement-list -> statement-list statement

statement-list -> @

statement -> expression-stmt

statement -> compound-stmt

statement -> condition-stmt

statement -> dowhile-stmt

statement -> return-stmt

expression-stmt -> expression SEMI

expression-stmt -> SEMI

condition-stmt -> if LLM expression RLM statement

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
condition-stmt -> if LLM expression RLM statement else statement
dowhile-stmt -> do statement while LLM expression RLM SEMI
return-stmt -> return SEMI
return-stmt -> return expression SEMI
expression -> variable ASSIGN expression
expression -> simple-expression
variable -> ID
variable -> ID LMM expression RMM
simple-expression -> additive-expression relop additive-expression
simple-expression -> additive-expression
relop -> LTEQ
relop -> LT
relop -> RT
relop -> RTEQ
relop -> EQ
relop -> NE
additive-expression -> additive-expression addop term
additive-expression -> term
addop -> PLUS
addop -> MINUS
term -> term mulop factor
term -> factor
mulop -> MULTIPLY
mulop -> DIVIDE
mulop -> MOD
factor -> LLM expression RLM
factor -> variable
factor -> call
factor -> NUMBER
call -> ID LLM arguments RLM
arguments -> argument-list
arguments -> @
argument-list -> argument-list DOU expression
argument-list -> expression
```

语义函数：

```
0
0 1
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
0
0
0
1 0 -1
1 0 -1 2 -1 -1
0
0
0
0
1 0 -1 2 -1 3
0
0
0 -1 1
0
1 0
1 0 -1 -1
-1 0 1 -1
0 1
-1
0 1
-1
0
0
0
0
0
0 -1
-1
0 -1 1 -1 2
0 -1 1 -1 2 -1 3
0 1 -1 -1 2 -1 -1
0 -1
0 1 -1
1 0 2
0
0
0 -1 1 -1
1 0 2
```

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

```
0
0
0
0
0
0
0
0 1 2
0
0
0
0 1 2
0
0
0
0
-1 0 -1
0
0
0
0 -1 1 -1
0
-1
0 -1 1
0
```

输入文法：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.2 测试结果：求解 first 集合

项目过多，在此不一一展示。

2.3 测试结果：求解 follow 集合

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

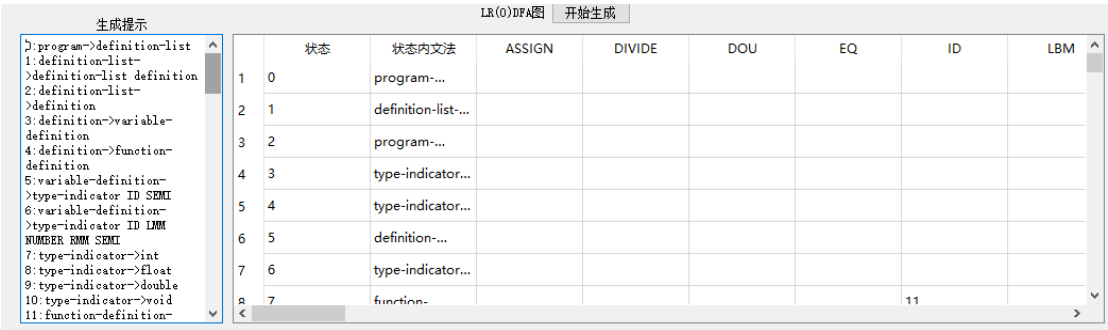
课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

项目过多，在此不一一展示。

2.4 测试结果：LR0



项目太多，未能展示完全

2.5 测试结果：SLR1 表

项目太多，未能展示完全
出现移进归约冲突，我们做只移进不规约的处理

2.6 测试结果：语法树代码生成

注意：lex 所在路径即为代码生成路径

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

2.7 测试结果：查看语法树生成代码文件

PS:请用 C++11 以上进行编译

华 南 师 范 大 学 实 验 报 告

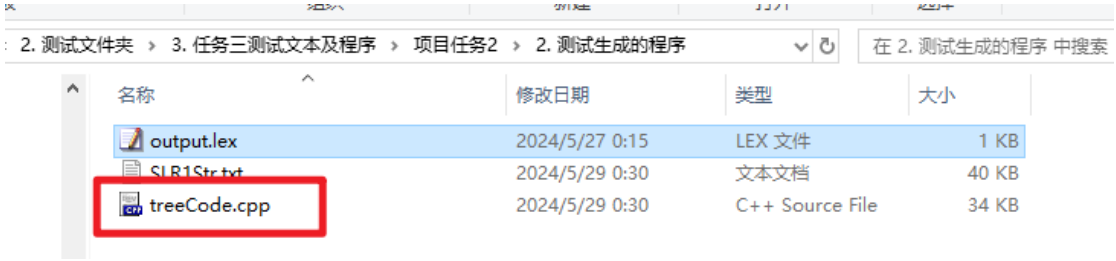
学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

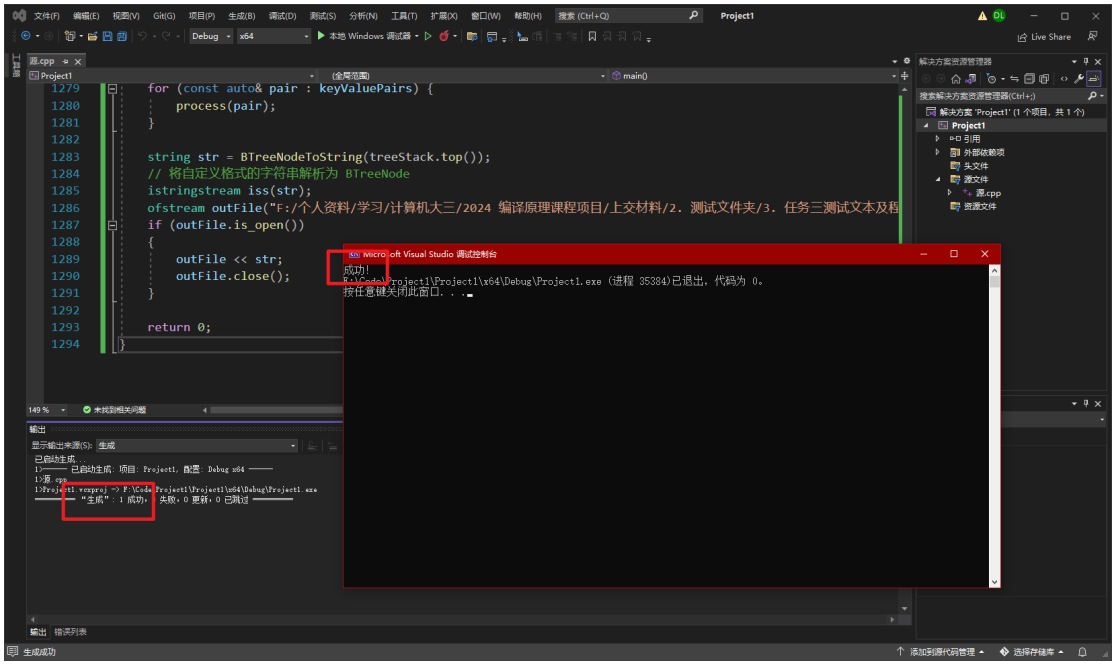
实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

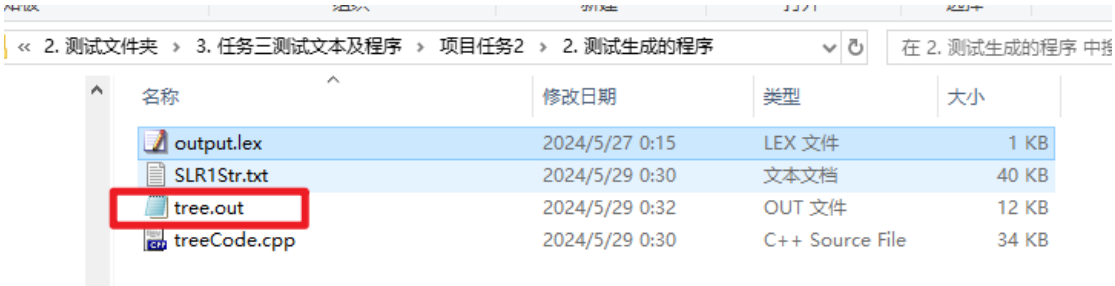


具体语法树代码太长，在此不进行展示，请打开 treeCode.cpp 查看

编译运行：



得到语法树：



2.8 测试结果：可视化语法树

华 南 师 范 大 学 实 验 报 告

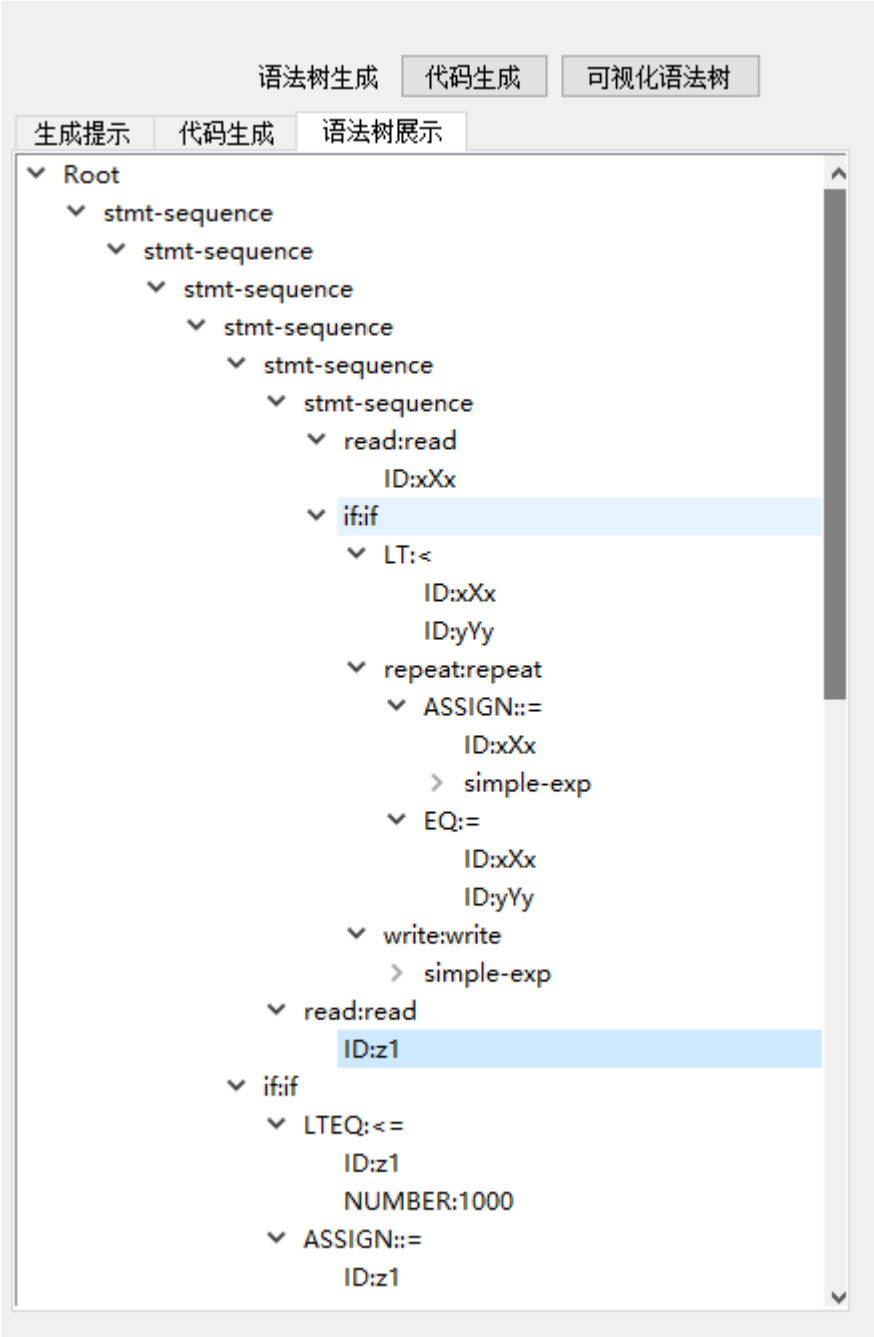
学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分



测试结果

项目任务三-项目二的测试完全通过

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

四、实验总结（心得体会）

1. 项目任务 1 的心得体会：

这个项目任务一涉及了从 NFA 到 DFA 的转换，然后再对 DFA 进行最小化，并最后将其用以生成词法分析代码，以用作简单的词法分析器。首先将一个给定的 NFA 转换为一个等效的 DFA。这一过程中使用了 NFA 的状态迁移、 ε -闭包等核心概念。在转换的过程中，通过逐步构建 DFA 状态和状态迁移表，将一个可能非确定性的自动机转化为确定性的自动机。一旦获得 DFA，实验继续进行 DFA 的最小化。最小化的目的是简化 DFA，去除不必要的状态，并保留其等效性。实验使用了等价关系分割状态空间的方法，以减少 DFA 的状态数量，从而减小内存和计算要求。这个实验有助于理解自动机理论和如何将其应用于编写词法分析器。同时，通过将自动机转换和最小化的步骤应用于实际问题，可以学到如何将理论知识转化为实际应用，生成可以处理一段程序的词法分析代码。为词法分析提供了一种简单而有效的解决方案，可以用于从输入程序中识别和提取特定的符号。

总的来说，这个项目任务一提供了一个完整的学习过程，涵盖了自动机理论、状态转移、最小化以及代码生成，为理解和应用自动机在编程中的重要性提供了实际示例。

2. 项目任务 2 的心得体会：

在本次项目任务中，我们成功设计并实现了一个用于处理文法的应用程序并成功生成了语法树。我们成功计算并呈现了文法中各非终结符的 First 集合与 Follow 集合。这些集合的计算对于语法分析表的构建非常关键，展示了文法对应的 LR(0) DFA 图，这有助于用户理解文法的状态转换和 LR(0)自动机的结构，并且程序还能判断文法是否为 SLR(1)文法。如果文法不是 SLR(1)文法，用户可以查看原因，这对于纠正文法错误或优化文法设计非常有帮助。在文法为 SLR(1)文法时，我

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学 号 20203231004

专 业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

们提供了一个窗口展示文法的 SLR(1)分析表。这个表格显示了文法符号、状态、动作等信息，帮助用户理解 SLR(1)分析过程。最后，我们生成了语法树程序的代码，用户通过语法树程序，可以得到某个高级语言程序的可视化语法树，对理解程序编译背后的原理有着极大的作用。

整体而言，项目任务 2 取得了成功。程序实现了任务书要求的各项功能，用户可以方便地输入、管理文法，查看 First 集合、Follow 集合、LR(0) DFA 图、SLR(1)文法判断、SLR(1)分析表的结果和某个高级语言程序相对应的语法树，从而更好地理解和分析文法。

3. 项目任务 3 的心得体会：

在第三个项目任务中，我们以 mini-c 语言作为测试对象，全面体验了词法和语法分析的完整流程。通过此次任务，我深刻理解了理论与实践结合的重要性，也体会到了软件开发过程中的挑战和成就。

首先，通过编写和输入 mini-c 的词法正则表达式，我们通过系统界面生成了相应的 NFA、DFA 和最小化 DFA 以及相应的词法分析程序，

其次，在语法分析部分，我们输入了 mini-c 的 BNF 文法，并通过系统计算出了每个非终结符号的 First 和 Follow 集合。随后，我们构建了 LR(0)DFA 图和 SLR(1)分析表。

最后，以自编写的 mini-c 源程序进行测试，我们成功地进行了词法分析，生成了单词编码文件，并基于此文件进行了语法分析，生成了对应的语法树。整个过程验证了系统的功能，也证明了我们对理论知识的应用能力。

五、参考文献：

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

《编译原理》课程课件

六、项目自评

总评自评：80 分

理由：

（1）项目任务一的全部项目内容都已经实现，能够打开某一高级语言的正则表达式，能够将用户输入的正则表达式文件转换成 NFA，NFA 转换成 DFA，将 DFA 最小化，将得到的最小 DFA 转换成词法分析程序，并提供窗口供用户查看，能够以该高级设计语言的一个源程序进行测试，输出该源程序词法分析后的单词编码，并提供窗口供用户查看该单词编码。对词法分析器进行测试，测试样例完整，结果符合预期。支持长正则表达式以及状态数很多的情况，并且 DFA 最小化结果完全正确。能完成 tiny 语言、minic 的词法分析程序生成。能够运用合理的数据结构帮助实现，系统的设计分析清晰。可视化界面简洁，人机交互性良好。

（2）项目任务二的必做项目内容都已经实现，能够打开某一高级语言的文法，能够将用户输入的正确求出 first 集合、follow 集合、LR0 DFA 图、SLR1 分析表和能够通过某一高级语言的语义函数文件、任务一的 lex 文件、SLR1 分析表生成语法树程序代码，并通过编译运行，得到某一高级语言编写程序的语法树，并提供窗口供用户查看。对本程序进行测试，测试样例完整，结果符合预期。支持各种语言的文法情况，并且 first、follow、LR0、SLR1 分析正确，语法树展示完整。能完成 tiny 语言、minic 的程序的语法树生成。能够运用合理的数据结构帮

华 南 师 范 大 学 实 验 报 告

学生姓名 李达良 学　号 20203231004

专　业 计算机科学与技术 年级、班级 2021 级 1 班

课程名称 编译原理项目 实验项目 编译原理项目

实验时间 2024 年 5 月 25 日

实验指导老师 黄煜廉 实验评分

助实现，系统的设计分析清晰。可视化界面简洁，人机交互性良好。

但遗憾的是，没有完成选做内容，但个人认为，其实实现原理就是将语法树生成代码进行改写，得到中间代码，由于时间和个人安排的关系，未能完成此任务。

（3）项目任务三以 mini-c 的词法进行测试，我们设计了一个 mini-c 源程序进行词法分析的测试并以生成源程序所生成的单词编码文件进行语法分析，生成对应的语法树，生成的单词编码文件准确，语法树展示完整，通过项目任务三的所有测试用例。