

[一、实验内容 3](#_Toc119248870)

[1.概述 3](#_Toc119248871)

[2.必做内容 3](#_Toc119248872)

[3.选做内容 3](#_Toc119248873)

[4.备注 3](#_Toc119248874)

[二、实验目的 4](#_Toc119248875)

[三、实验过程 4](#_Toc119248876)

[（一）理论准备 4](#_Toc119248877)

[0.数据结构 4](#_Toc119248878)

[1.化简文法 4](#_Toc119248879)

[2.求出first集合 5](#_Toc119248880)

[3.求出follow集合 5](#_Toc119248881)

[4.消除左公因子 7](#_Toc119248882)

[5.消除左递归 7](#_Toc119248883)

[6.生成自动机 8](#_Toc119248884)

[（二）代码架构 8](#_Toc119248885)

[1.界面与文件存取 8](#_Toc119248886)

[2.文法处理器 9](#_Toc119248887)

[3.自动机 9](#_Toc119248888)

[（三）代码细节 9](#_Toc119248889)

[（四）测试数据 9](#_Toc119248890)

[（1）第一组测试数据 9](#_Toc119248891)

[（2）第二组测试数据 9](#_Toc119248892)

[（五）存在问题 9](#_Toc119248893)

[四、实验总结 10](#_Toc119248894)

[（一）学习感悟 10](#_Toc119248895)

[（二）编码心得 10](#_Toc119248896)

[五、参考文献 10](#_Toc119248897)

# 一、实验内容

## 1.概述

设计一个应用软件，以实现文法的化简及各种问题的处理。

## 2.必做内容

（1）系统需要提供一个文法编辑界面，让用户输入文法规则（可保存、打开存有文法规则的文件）

（2）化简文法：检查文法是否存在有害规则和多余规则并将其去除。系统应该提供窗口以便用户可以查看文法化简后的结果。

（3）检查该文法是否存在着左公共因子（可能包含直接和间接的情况）。如果存在，则消除该文法的左公共因子。系统应该提供窗口以便用户可以查看消除左公共因子的结果。【从某个非终结符号开始只需要消除做4次推导内就可以发现的左公共因子，如果超过4次推导还存在左公共因子就直接产生出错处理即可】

（4）检查该文法是否存在着左递归（可能包含直接和间接的情况），如果存在，则消除该文法的左递归。系统应该提供窗口以便用户可以查看消除左递归后的结果。

（5）求出经过前面步骤处理好的文法各非终结符号的first集合与follow集合，并提供窗口以便用户可以查看这些集合结果。【可以采用表格的形式呈现】

（6）如果输入的文法规则是线性规则（左线性规则或右线性规则），则将该文法对应的有穷自动机生成出来，系统应该提供界面让用户可以查看转换得到的有穷自动机（可以使用状态表的方法来呈现）。【注意：如果该有穷自动机是NFA，还需要把该NFA转为DFA并呈现，并再进行DFA的最小化并呈现，你可以使用实验二的算法来完成】

（7）软件的操作界面应该是windows界面

（8）应该书写完善的软件文档

## 3.选做内容

（1）对于必做内容中的第（6）个要求，使用画图的算法设计把DFA图，NFA图等直观图示呈现在窗口中。

## 4.备注

（1）注意事项：文法规则为了处理上的简单，输入时可以只输入单一个大写字母作为非终结符号，单一个小写字母作为终结符号，用@表示空串。

如，可以这样输入：

E->E+T

T->a

（2）补充说明：4次推导才发现左公共因子的举例：

S->Aa | Ba

A->Cb

B->Dc

C->Ed

D->Fe

E->fe

F->fd

如果超过4次才发现的左公共因子的就产生出错，如果少于等于4次推导的就要把左公共因子进行消除。

# 二、实验目的

（1）理解文法问题的解决算法

（2）加深对状态机的NFA图、DFA图的理解

（3）增强编码能力

# 三、实验过程

## （一）理论准备

### 0.数据结构

（1）定义

文法：描述语言语法结构的一系列形式规则。

上下文无关文法CFG是一个四元组G=（VN非终结符的集合，VT终结符的集合，P产生式的集合，S开始符号）。

整个实验都是在分析和重构文法。那么就要尤其注意**存储**文法和**分析**文法的数据结构。

（2）存储结构

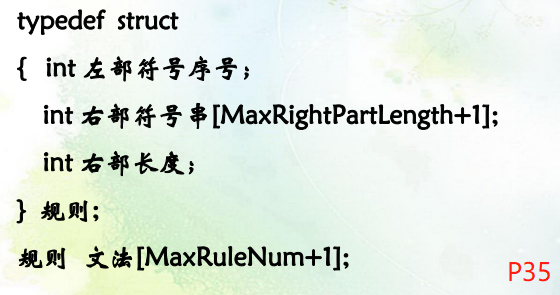
文法规则有数组和链表两种结构，我选择链表。

首先，用一个**字典**存储读入的字符并为其安排对应的序号。终结符号的序号大于1小于100，非终结符号的序号大于100。

|  |
| --- |
| map<char, int> v; |

接着，定义一个**结构体**，名为“规则”，参考：

|  |
| --- |
| struct Rule{  int leftChar;  vector<int> rightChar;  };  List<Rule> grammar; |



// vector适合随机存取，list适合插入和删除

为了便于将来的操作，需要为非终结符排序。

|  |
| --- |
| list<Rule> vn; |

另外，可以为它定义一个**打印函数**PrintGrammar()，打印A->a格式的字符串。

（3）分析结构

在“规则”结构体的基础上，构建**文法链表**。

|  |
| --- |
| list<Rule> derivation; |

分析的过程又是推导的过程，并选用最左推导。

### 1.化简文法

化简文法就是去除其中的无效规则，其中无效规则包括了有害规则和多余规则。

（1）有害规则

定义：导致文法出现二义性的规则。例如A->A，B->B。

算法：直接找左部符号和右部符号一样的文法规则结构体，删了。【比较】

（2）多余规则

定义：文法中任何句子的推导都不会用到的规则，又分为不可到达规则和不可终止规则。

算法：

①去除不可到达规则：

将S放入“可到达集合”中。从S开始遍历每个文法，若左部在集合中，则将右部的每个非终结符也放入“可到达集合”中。再次遍历每个文法，若左部不在集合中，则删去。【并查集】

②去除不可终止规则：

【深度优先搜索】【递归】

①从开始符号起步，遍历每个非终结符，遍历其对应的每个文法，检查是否满足A->αAβ形式。

②若满足，则遍历每个以A为左部的文法，检查是否存在不满足A->αAβ形式的文法。若存在，则结束遍历，不做操作。若不存在，则删除该文法。

③若不满足，且右部存在非终结符B，则遍历每个以B为左部的文法。若存在以B为左部的文法，则重复①。若不存在，则删除该文法。

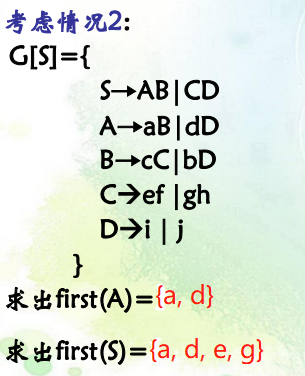
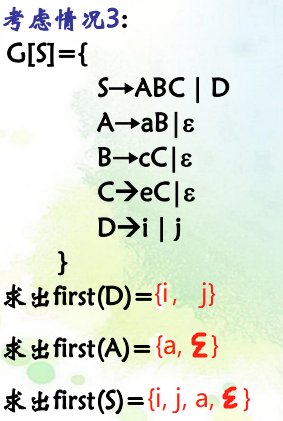
④若不满足，且右部不存在非终结符，则无操作。

举例：S->Be，B->Ce，B->Af，A->e ，A->Ae，A->E，E->E ，C->Cf，D->f。

### 2.求出first集合

（1）定义

可以从**非**终结符号或符号串X**推导出**的所有串首**终结符**构成的**集合**。如果X->∗ε，那么 ε也在FIRST（X）中。举例：

（2）手工算法

以X为目标，从右部符号串取第一个字符a，若a是终结符或ε，则填入Follow集中；若a是非终结符，则继续重复上述步骤，以a为左部，向右推导，直到其右部符号串的**串首**是终结符或ε。

（3）代码算法

定义【递归】函数GetFirst(int X)。遍历以X为左部的文法，取文法右部第一个字符假设为a。若a是终结符或ε，则填入当前Follow集中；若为非终结符，则递归进入GetFirst(int a)。遍历结束则返回。

### 3.求出follow集合

（1）定义

给定一个在右部的非终结符A，则Follow(A)为**紧跟**在其后的每个**终结符号**或$（右端结束标记）的**集合**。举例：

|  |  |  |
| --- | --- | --- |
| E -> TE’  E’-> +TE’| ε  T -> FT’  T’-> \*FT’| ε  F -> (E) | a  Follow(E)={ $, ) }  Follow(E’)={ $, ) }  Follow(T)=  { +, $, ) }  Follow(T’)=  { +, $, ) }  Follow(F)=  { \*, +, $, ) }  (来源：[编译原理中Follow集的求法\_杨博东的博客的博客-CSDN博客\_follow集](https://blog.csdn.net/yangbodong22011/article/details/52950436?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-52950436-blog-106379883.pc_relevant_recovery_v2&spm=1001.2101.3001.4242.1&utm_relevant_index=3)) | S→ABc  A→a|ε  B→b|ε  Follow（S）=  ｛＃｝  Follow（A）=  ｛b，c｝  Follow（B）=  ｛c｝  （来源：[编译原理 First集 Follow集 select集 通俗易懂的讲解 + 实例\_CooperNiu的博客-CSDN博客\_select集](https://blog.csdn.net/CooperNiu/article/details/78524688?spm=1001.2101.3001.6650.6&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-6-78524688-blog-115911002.pc_relevant_aa2&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-6-78524688-blog-115911002.pc_relevant_aa2&utm_relevant_index=7)） | S→AB  S→bC  A→ε  A→b  B→ε  B→aD  C→AD  C→b  D→aS  D→c  FOLLOW(S)={ $ }  FOLLOW(A)= { a, $, c }  FOLLOW(B)= { $ }  FOLLOW(C)={ $ }  FOLLOW(D)={ $ }  （来源：[判断LL(1)文法（first集、follow集、select集）\_内存不足°的博客-CSDN博客\_判断ll(1)文法](https://blog.csdn.net/qq_44922497/article/details/111880076?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-4-111880076-blog-78524688.pc_relevant_recovery_v2&spm=1001.2101.3001.4242.3&utm_relevant_index=7)） |

（2）手工算法

不断应用下面的规则，直到再没有新的终结符号可以被加入到任意的follow集合中为止：

①将 **$** 放到follow（S）中，其中S是文法的开始符号。

②当A是最右部的时候，将 **$** 加入到follow(A)中

③如果存在一个产生式A→αBβ，那么follow（B）**包含first（β）-ε**。（ follow(B)是求跟在B后的终结符或$组成的集合，因此对于跟在B后的β，它的first集合就是follow(B)的子集 ）

④如果存在一个产生式A→αB，或存在产生式A→αBβ且first（β）包含ε，那么follow（B）**包含follow（A）**。（对于A→αBβ,且β多步推导出ε ，那么可以用αB替换A, B后面紧跟的字符就是A后面紧跟的字符）

（3）代码算法

定义【递归】函数GetFollow(int X)。

若X是文法开始符号，则将$填入当前Follow集中。遍历每个文法。

若X在文法最右部，则将$填入当前Follow集中。

若存在A->αXβ形式，则将GetFirst(β)-ε填入当前Follow集中。

若GetFirst(β)包含ε，则将GetFollow(A) 填入当前Follow集中。

若存在A->αX形式，则将GetFollow(A) 填入当前Follow集中。

### 4.消除左公因子

（1）定义

一个或多个文法规则共享一个通用前缀串。

举例：A->ab，A->ac。B->acmm，B->acd。

（2）手工算法

提取左公因子，将后缀改为“左部符号**’**”，并新增文法：左部符号’->后缀

举例（承接上面的例子）：A->aA’，A’ ->b | c。B->acB’，B’ ->mm | d。

（3）代码算法

定义【迭代】函数Remove\_ Left\_ Common\_Factor ()。

①遍历每个非终结符号，设当前符号为Vni

②遍历以Vni为左部的文法，装入容器中并统计个数j。

③若个数j≠1，则定义字典Map<vector, int>，和数组union[ j ]（初始化为0）。遍历文法，若右部第一个字符为非终结字符A，则在字典添加First(A)；若为终结字符a，则在字典添加{a}，并分别赋序号m。若字典已有某个集合，且其对应的序号为n，则union[m]=n。【并查集】

④遍历union，若发现union[m]≠0而＝n，找到第n个文法，其原形式为A->aB，新增非终结符C，新增文法A->aC，C->B，删除原文法。找到所有union[…]=n的第…个文法A->aB’，新增文法C->B’。

### 5.消除左递归

（1）定义

左递归：文法经过一次或多次推导之后，出现如下形式A->Aα，则称该文法是左递归的。左递归会产生回溯。

①直接左递归：经过一次推导就可以看出文法存在左递归。如A->Aα|β。递归结果为A->Aα->Aαα->Aααα（总在左边增加α）->βααα= β{α}

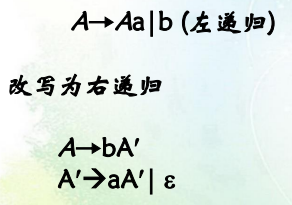
②间接左递归：需多次推导才可以看出文法存在左递归。如文法：S->Qc | c，Q->Rb | b，R->Sa | a，有S->Qc->Rbc->Sabc。

（2）手工算法

①消除直接左递归：

举一个最简单的例子，把A->Aa | β直接改为A->βA’，A’ ->aA’ | ε（右递归）。

更一般化的形如P → P**X** | **Y**（其中X和Y看作一个整体，比如：P → Pabc|ab|b，X就是abc，Y就是ab|b），可改写为P->**Y**P'，P'->**X**P' | **ε**。

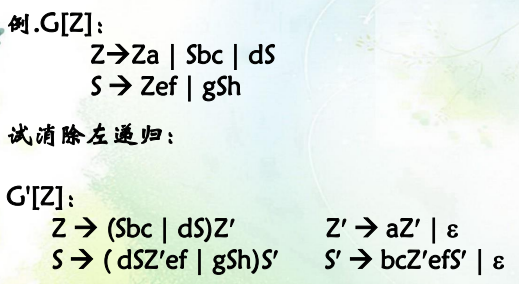


②消除间接左递归：

1）若消除过程中出现了直接左递归，则按照左递归的方法直接消除

2）将消除直接左递归后的新文法代入未解决的文法中（即间接左递归），得到新的直接左递归，按照步骤1再次消除

3）反复实施，直到不可代入



（3）代码算法

定义【迭代】函数Remove\_ Left\_Recursion()。

①遍历每个非终结符号，设当前符号为Vni

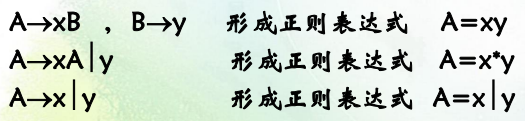
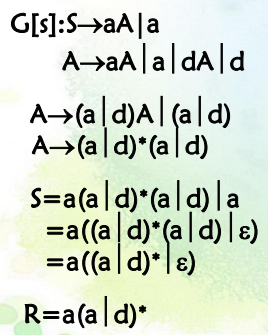
②遍历该符号之前的非终结符号Vnj

③遍历以Vnj为左部的文法，**代入**到形式为Vni→αVnj β的文法中，得到新的Vni的文法

④判断Vni是否存在形式为A→ A**X**的文法。若存在，则新增非终结符B，新增文法B->XB，和文法B->ε，删除原文法。遍历其它以Vni为左部的文法，将B追加到最右部。（**消除直接左递归**）

### 6.生成自动机

（1）将文法规则转换为正则表达式

①遍历每个非终结符A->……，形成正则表达式A=……

②处理S，将各表达式代入到S的表达式中

（2）生成NFA图

（3）生成DFA图

（4）生成最小DFA图

## （二）代码架构

### 1.界面与文件存取

#### （1）界面设计

#### （2）文件读取

定义一个Loader类，**专门**作为读取文件的工具，因此将其设为**单例**。C++的饿汉式单例模式实现起来非常简单粗暴，用静态变量和静态函数即可。但存在线程安全问题，但不在本次作业的考虑范围内。

|  |
| --- |
| class **Loader**  {  private:  static char\* filePath;  public:  static char\* **Input**(const char\* filePath);  static void **Output**(const char\* compressed);  static char\* **GetFilepath**();  }; |

#### （3）QString转char

|  |
| --- |
| QByteArray fn\_qba = fileName.toLatin1();  char\* fn\_cc = fn\_qba.data();  char\* c = Loader::Input(fn\_cc); |

#### （4）char转QString

|  |
| --- |
| QString(c) |

### 2.文法处理器

定义一个类专门负责处理文法规则，名为GrammarProcessor。

#### （1）数据结构

参考理论准备部分，

#### （2）基础工具代码

根据理论准备部分的分析，可知以下函数需要经常被用到：

①bool MatchFormat(Rule r, bool alpha, bool B, bool beta)

判断文法r是否满足格式A->[α][B][β]，三者皆为可选。

②

#### （3）去除有害规则

定义函数RemoveHarmfulRules()

#### （4）去除不可达规则

#### （5）去除不可终止规则

#### （6）求First集合

#### （7）求Follow集合

#### （8）消除左公因子

#### （9）消除左递归

### 3.自动机

定义一个负责转化表达式、NFA、DFA的自动机类，重写实验2的RegularExpression2NFA类的代码，名为Automata。

（1）文法规则转表达式

（2）

## （三）代码细节

## （四）测试数据

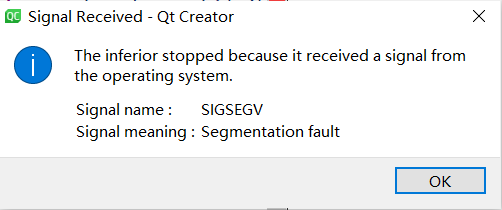
一个数据文件涉及的处理过程足够繁杂，且本人能力有限，再次仅设计2组测试数据。

### （1）第一组测试数据

### （2）第二组测试数据

## （五）存在问题

（1）



老师说，没遇到过段错误，都不能说自己敲的代码多。

Debug之后发现是指针问题，把**新开辟**的数组赋值给了错误的变量a，然后再把这个变量a赋值给别的变量b，就出问题了。

已修复。

# 四、实验总结

## （一）学习感悟

## （二）编码心得

# 五、参考文献

（1）[(2条消息) 包教包会！十分钟搞定自顶向下分析——编译原理速成计划\_smilestruggler的博客-CSDN博客\_编译原理速成](https://blog.csdn.net/toohandsomeIeaseId/article/details/105010226)

（2）