**深 圳 大 学 实 验 报 告**

**课程名称：­ 编译原理**

**实验项目名称： 高级语言及其文法**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 张昊迪**

**报告人：缪克达 学号：2020274045 班级： 二班**

**实验报告提交时间： 2023年3月16日星期四**

**教务处制**

1. 实验内容

文法（Grammar）是描述高级语言语法结构的重要工具。定义任意的文法 G， 需要完成对其四元组（V,T,P,S）的定义（课本 P33）。在该实验中，请制定文法 文件的具体组织形式、编程完成对文法文件的读取、并完成对文法的分类。该实 验具体包含以下两个任务：

* 任务一：文法的定义及读取

现规定文法由 Grammar.txt 文件保存，请制定文法文件的具体存储格式。如 文法 G={{S,A,B,C}, {a,b,c}, {S->ABC,A->a,B->b,C->c}, S}在Grammar.txt 文件 中可由以下方式描述并存储：

--------------------------------------------------

S,A,B,C

a,b,c

S->ABC,A->a,B->b,C->c

S

--------------------------------------------------

文法的文本形式可根据自己需要自由定义，在此基础上，编程实现对任意文法文件的读取。

* 任务二：文法的分类

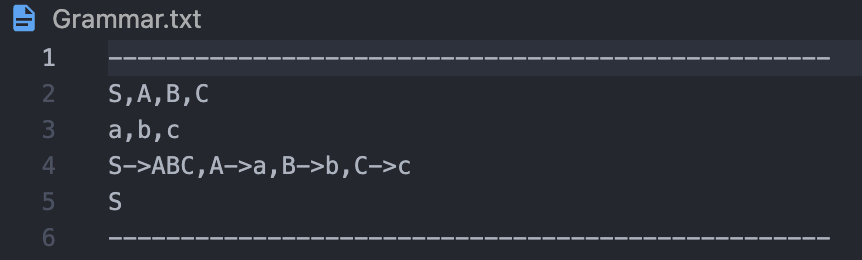
根据 Chomsky 的文法体系分类，文法分为四大种类。请在任务一的基础上，编程实现对 Grammar.txt 中存储的文法进行分类，自动判断其所属类别。例如任务一中所给出的文法 G 应被判定为 2 型文法，即上下文无关文法。请设计分类方法，并设计四类不同的测试文法测试分类结果的正确性。

1. 实验要求

* 使用任意语言完成任务一、二的程序编写；
* 使用实验所提供的模板撰写实验报告，要求内容详实，有具体的设计描述、 关键的代码片段、及实验结果屏幕截图；
* 在截止日期前将代码、实验报告、测试文件等所有实验相关文件压缩到一个 压缩包姓名\_学号\_实验一.zip 上传至 Blackboard。

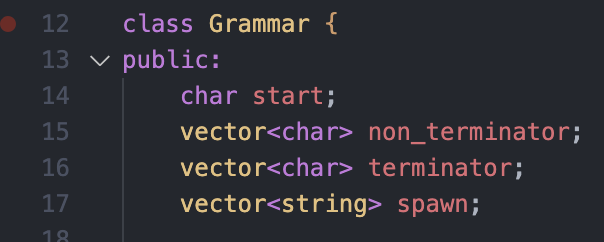
1. 实验步骤与结果
2. 任务1

任务1要求我们通过读取文法的格式，然后判断出终极符、非终极符、开始符号和产生式。所以，假如Grammar.txt存在文法的定义，比如：



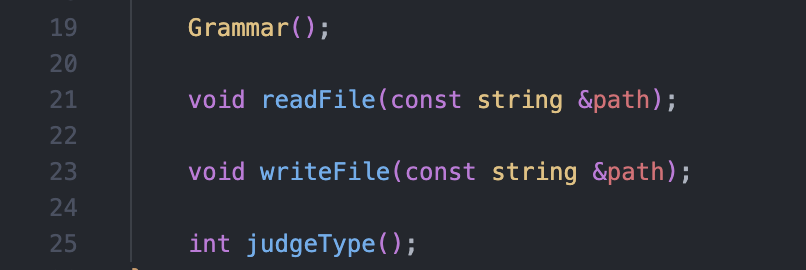
我们就需要读取这个文件，并得到相应的终极符、非终极符、开始符号和产生式，最后并加入到这个txt的末尾中，同时程序存储终极符、非终极符、开始符号和产生式的具体内容。本次实验我们采用C++。

首先定义一个Grammar类表示文法类：



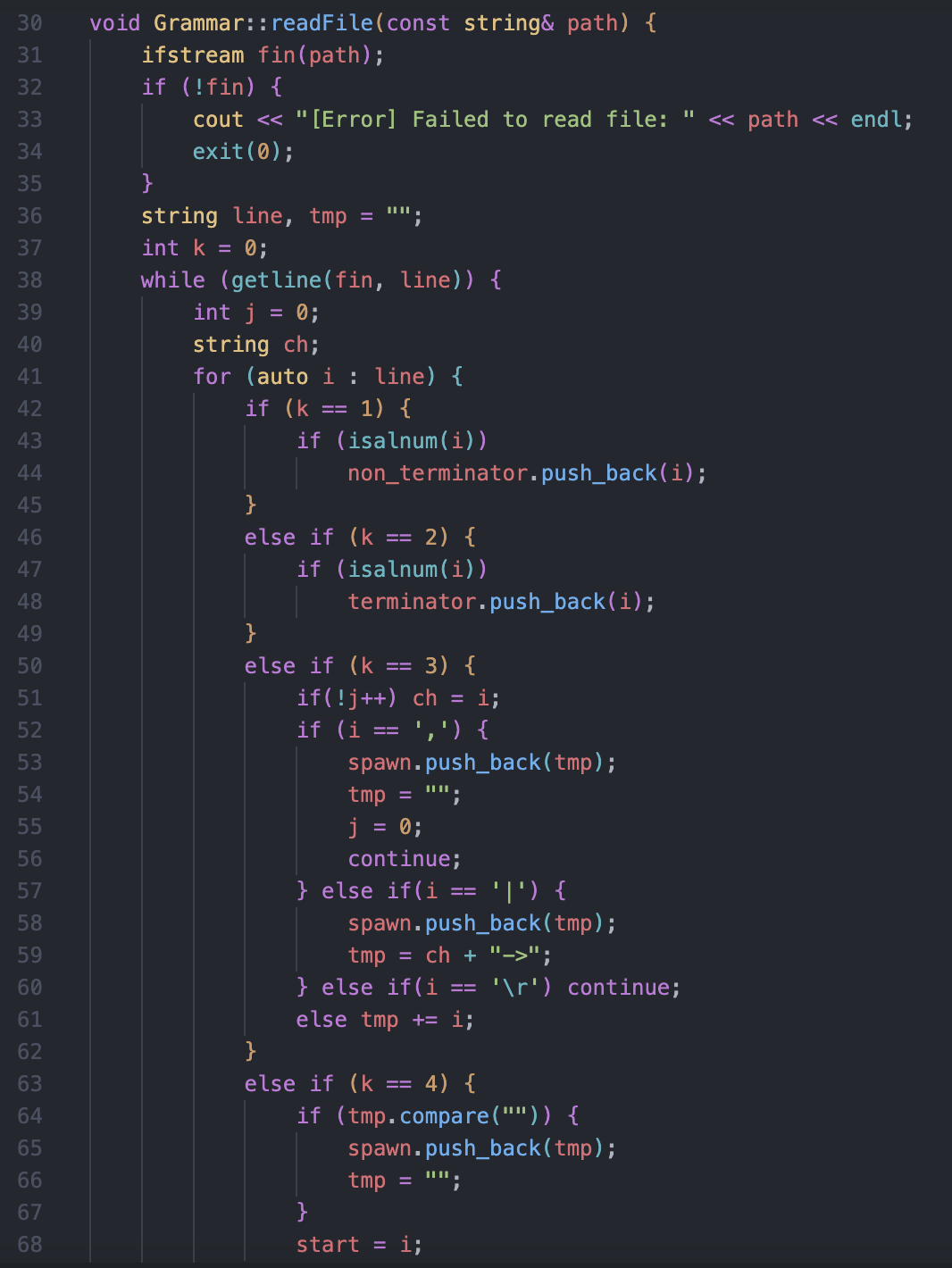
变量start是开始符号，non\_terminator是非终极符的集合，terminator是终极符的集合，spawn是产生式的集合。

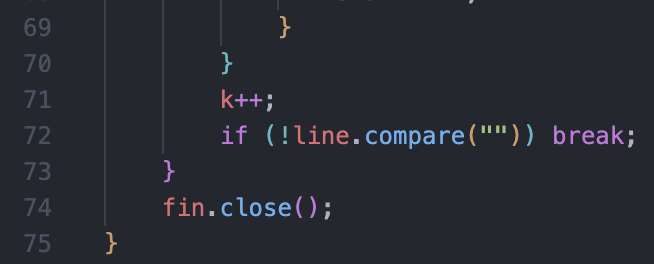
还有如下函数：



Grammar是构造函数，readFile是读取Grammar.txt的函数，writeFile是写入文本的函数，judgeType是判断文法类型的函数，它将返回文法的类型。

任务1中我们需要读取Grammar.txt，并获得终极符、非终极符、开始符号和产生式的具体内容。所以我们写下如下的内容：





具体来说，该函数读取了一个给定路径下的文件，打开并读取其中的内容。如果打开文件失败，函数将输出一个错误信息并退出程序。

然后，该函数按照特定格式解析文件内容，并将解析结果存储在对象的成员变量中。具体来说，该函数将文件内容划分为五个部分：

非终结符

终结符

产生式

起始符号

这段代码输入参数为一个字符串 path，表示要读取的文件路径。函数主要功能是读取文件并将文件中的语法规则保存到类成员变量中。

首先，函数打开文件，使用 ifstream 类型的对象 fin 对文件进行读取。如果文件打开失败，则输出错误信息并退出程序。

接下来，函数使用 getline 函数逐行读取文件内容，并使用 line 字符串变量保存每一行的内容。在每一行中，函数根据 k 的不同情况进行不同的处理。

当 k 为 1 时，函数将该行中的非终结符保存到类成员变量 non\_terminator 中。如果当前字符是大小写字母，则将其添加到 non\_terminator 中。

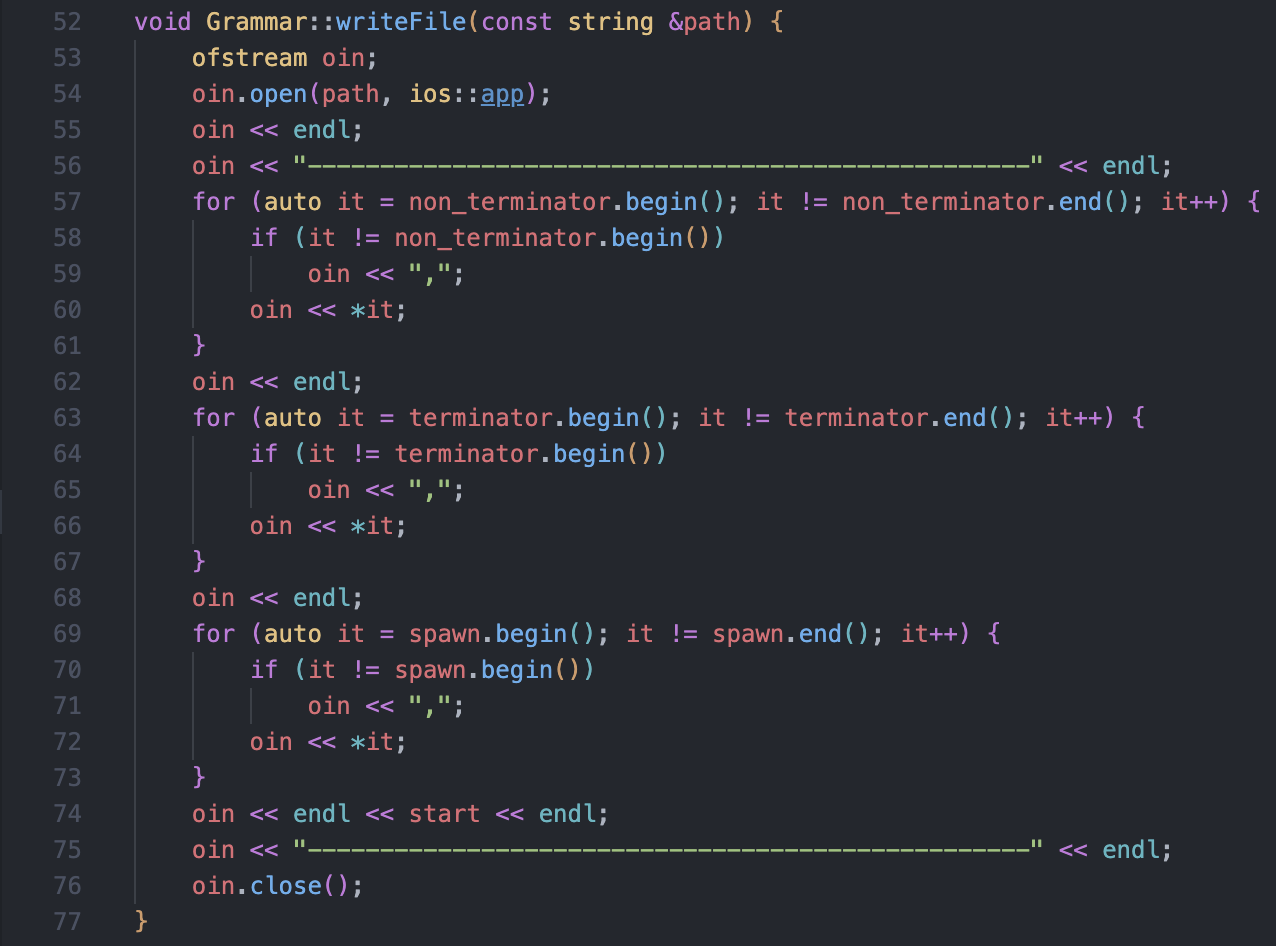
当 k 为 2 时，函数将该行中的终结符保存到类成员变量 terminator 中。如果当前字符是大小写字母，则将其添加到 terminator 中。

当 k 为 3 时，函数将该行中的产生式保存到类成员变量 spawn 中。如果当前字符是逗号，则说明当前产生式已经读取完毕，将之前保存的产生式字符串 tmp 添加到 spawn 中，并将 tmp 清空；否则将当前字符添加到 tmp 中。

当 k 为 4 时，函数将该行中的起始符号保存到类成员变量 start 中。

最后，函数将 k 加 1，用于处理下一行的内容。如果当前行的内容为空，则退出循环。最后关闭文件。

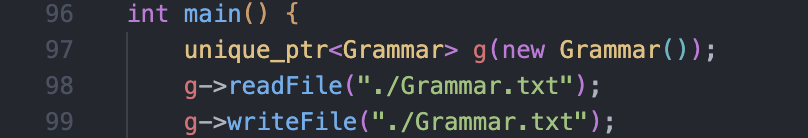
需要注意的是，代码中的 non\_terminator、terminator 和 spawn 都是 vector 类型的成员变量，分别用于保存非终结符、终结符和产生式。而 start 则是一个字符类型的成员变量，用于保存起始符号。读取之后，我们进行写入，如图所示：



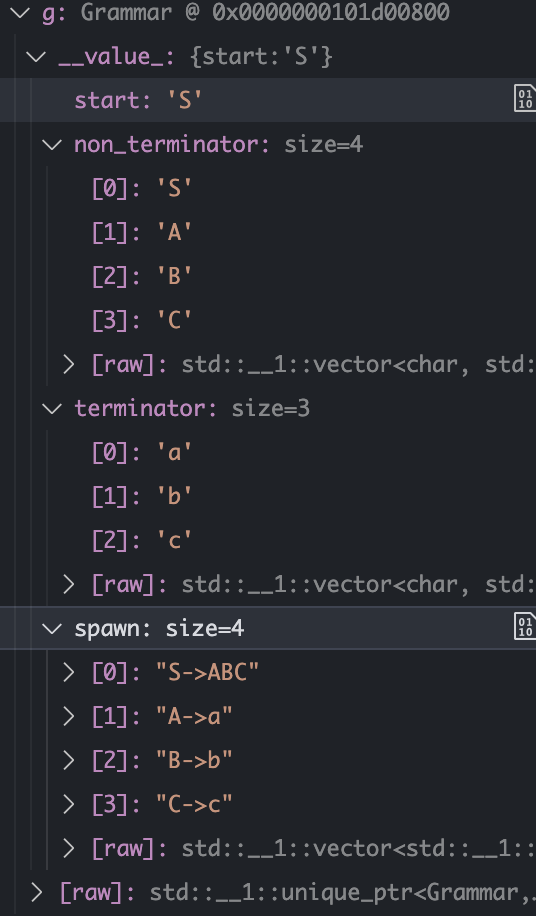
在写入过程中，该函数遍历了non\_terminator、terminator和spawn这三个容器，并将它们的内容依次写入到文件中。对于每个容器中的元素，该函数将其写入到文件中，并在元素之间添加逗号分隔符。为了保证输出格式的美观，该函数还在文件的开头和结尾添加了一些特定的分隔符和空白符。

最后，当所有的语法信息都被写入到文件中后，该函数将关闭文件，并返回操作结果。值得注意的是，该函数在打开文件时使用了ios::app参数，以便将新的语法信息附加到文件末尾而不是覆盖原有内容。

我们来进行验证，主函数如下：



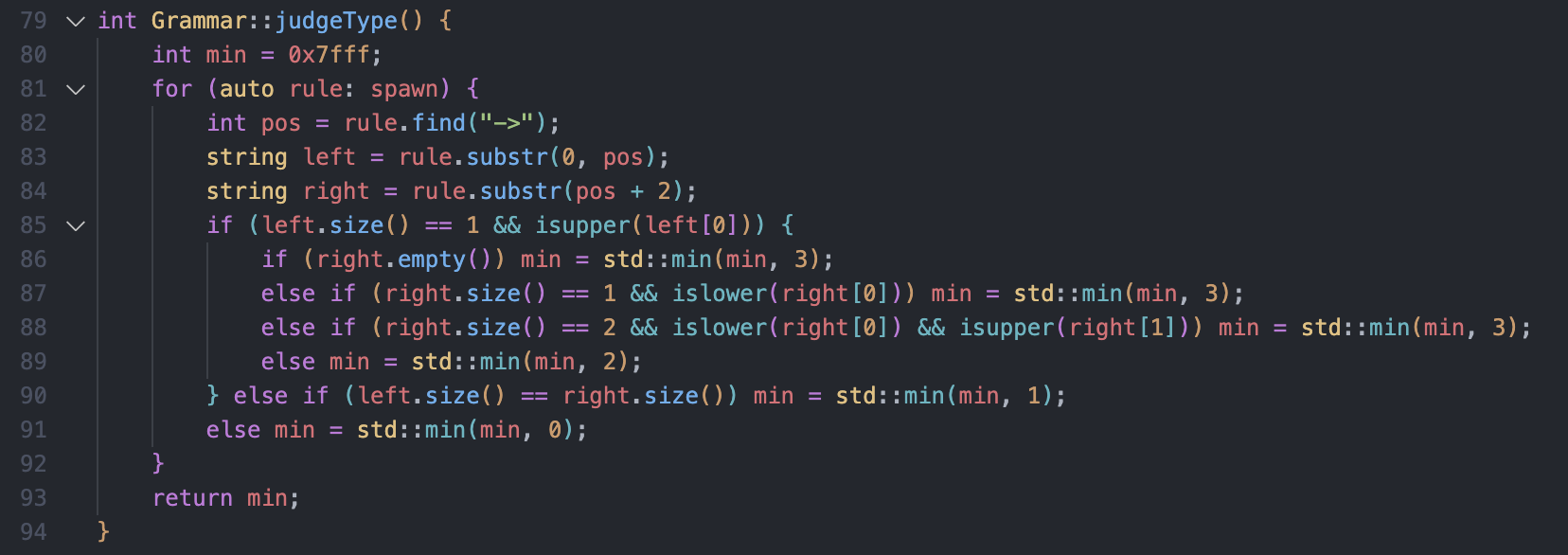
然后运行后变量如图所示：



可以发现我们成功读取了文法文件，验证成功！

1. 任务2

对于判断文法类型，我们一般从3类开始往下判断，具体代码如下：



这个函数是用于判断该文法属于哪种类型的上下文无关文法。它基于一些规则和算法，对文法中的每个产生式进行分析，计算出其所属的上下文无关文法类型，并返回最小的类型。

该函数首先将spawn中的每个产生式进行遍历，对于每个产生式，它都会将其分成左部和右部两个部分。然后，根据左部和右部的特征，判断该产生式所属的上下文无关文法类型。

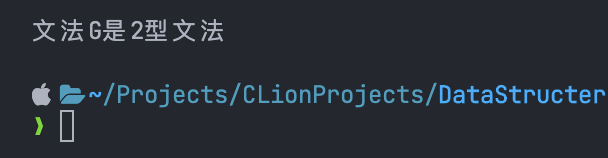
具体来说，该函数首先检查左部是否为单个非终结符，如果是，则对右部进行进一步的分析。如果右部为空，或者是一个单个小写字母终结符，或者是一个长度为2且第一个字符为小写字母，第二个字符为大写字母的组合，则该产生式属于上下文无关文法类型3，否则属于类型2。

如果左部和右部长度相等，则该产生式属于上下文无关文法类型1，否则属于类型0。对于所有的产生式，该函数会记录下其中最小的类型，并返回该值。如果所有产生式都无法满足上下文无关文法类型的要求，则返回类型0。

我们来验证下结果：



验证输出：

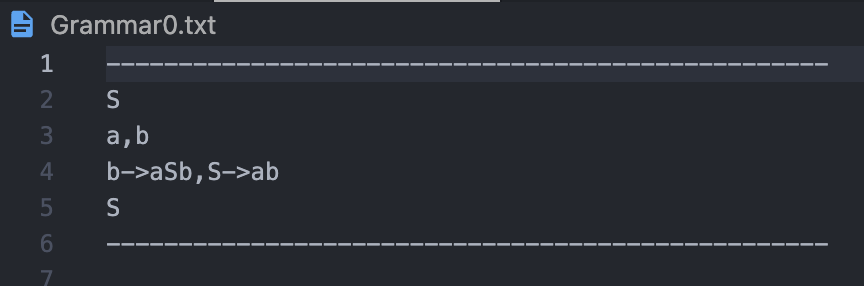


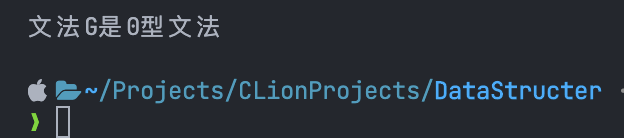
所以这个结果是对的。

1. 验证其他类型的文法类型：

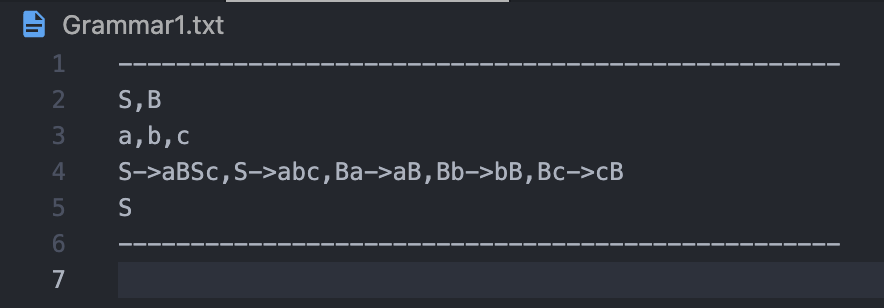
前面我们验证了2型文法，所以接下来我们验证0、1、3型文法。

0型文法：



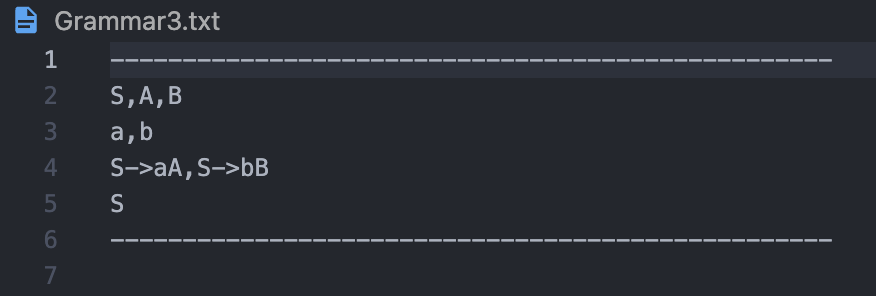


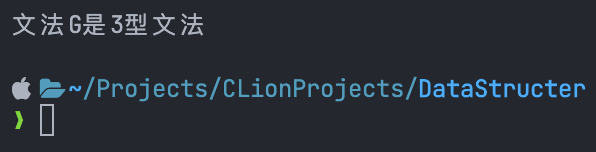
1型文法：





3型文法：





1. 实验总结

当我们处理自然语言时，文法是非常重要的工具。了解文法可以帮助我们理解自然语言的语法结构，从而更好地处理自然语言文本。在这个实验中，我们学习了文法的定义和分类，并实现了对任意文法文件的读取和自动分类。

在任务一中，我们制定了文法文件的具体存储格式，并实现了对任意文法文件的读取。具体而言，我们使用了 C++ 编程语言，并利用文件输入输出流实现了读取文法文件的功能。

在任务二中，我们使用了 Chomsky 的文法体系来对文法进行分类。具体而言，我们根据文法的规则、符号和起始符号的不同，将文法分为四类：0型文法（无限制文法）、1型文法（上下文有关文法）、2型文法（上下文无关文法）和3型文法（正则文法）。我们编写了一个 C++ 程序来自动判断给定的文法属于哪一种类型。

为了测试我们的程序，我们设计了四个不同的测试文法，分别属于四种不同的文法类型。我们将这些测试文法存储在不同的文本文件中，并用我们的程序对其进行分类。结果表明，我们的程序能够正确地判断给定文法的类型。

总之，这个实验让我们深入了解了文法的定义和分类，掌握了文法的读取和自动分类的技术，并加深了我们对自然语言语法结构的理解。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  年 月 日 |
| 备注： |