**编译原理与设计**

**实验报告**

实验名称： Lab 4：C语言语法文法设计与验证实验

1. **实验目的和内容**

**实验目的：**

本次实验的主要目的是了解程序设计语言的演化过程和相关标准的制定过程，深入理解与编译实现有关的形式语言理论，熟练掌握文法及其相关的概念，并能够使用文法对给定的语言进行描述，为后面的词法分析和语法分析做准备。

**实验内容：**

本次实验需要依次完成以下三项内容：

（1）阅读附件提供的 C 语言和 Java 语言的规范草稿，了解语言规范化定义应包括的具体内容。

（2）选定 C 语言子集，并使用 BNF 表示方法文法进行描述，要求至少包括表达式、赋值语句、分支语句和循环语句；或者设计一个新的程序设计语言，并使用文法对该语言的词法规则和文法规则进行描述。

（3）根据自己定义的文法子集，推导出“Hello World”程序。

以上语言定义首先要给出所使用的字母表，在此基础上使用 2 型文法描述语法规则。

具体步骤如实验指导文档所示。

1. **实验环境**

设备：RedmiBook 14 锐龙版

操作系统：Windows 10 Pro, 64-bit (Build 19045.2604) 10.0.19045

Java：openjdk version "1.8.0\_312"

ANTLRWorks：antlrworks-1.5.1(including ANTLR 3.5rc2)

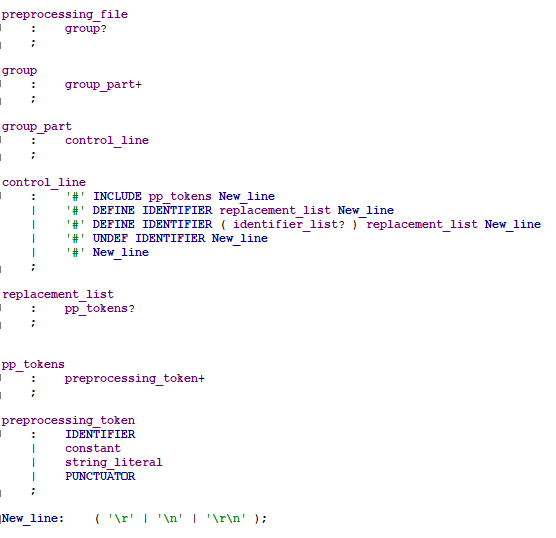
1. **实现的具体过程和步骤**

按照实验文档的指示：本次实验提供 C 语言和 Java 语言的规范草稿，请大家参考相关内容完成语言的设计和定义工作。此外，可以使用 ANTLRWorks 辅助完成语法的设计工作。

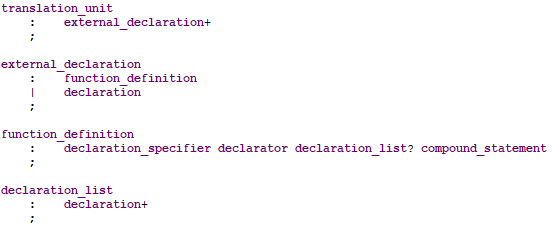
通过对C语言和Java语言规范的阅读，了解到语言规范应具备语言的字符集、词法规则、语法规则、语义规则，此外还可以包括程序结构、编译过程、标准库函数等内容。语言规范给编译器提供了统一的设计标准，不同实现方法的编译器才可以根据源代码编译相同功能的程序。语言规范同时也使得所有程序设计者可以基于同一标准设计程序，减小分歧，方便代码的共享与理解。

以下将使用 ANTLR 文法描述 C 语言的部分语法规则。 ANTLR 输入文法也是一种扩展的 BNF 表示方法。对于词法分析，沿用 lab3 的实验结果。而语法分析将分为多个部分，包括定义、声明、语句、与表达式。

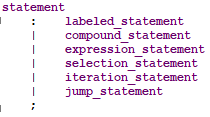
首先，指定终结符。语法分析的终结符一共有5种：关键字 KeyWord /标识符 Identifier /字符串字面量 StringLiteral /常量 Constant /分隔符及运算符 Punctuator，即词法分析得到的5种token。其余符号则皆为非终结符。

1. 预处理指令部分（Preprocessing directives），定义如下：

这一部分包含了预处理指令部分的文法，所有的预处理命令都由‘#’开头,并独占一行。包括宏定义、取消宏定义、包含文件命令等语句。

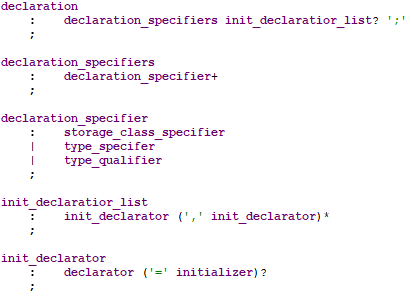
1. 外部定义部分（External definitions），定义如下：

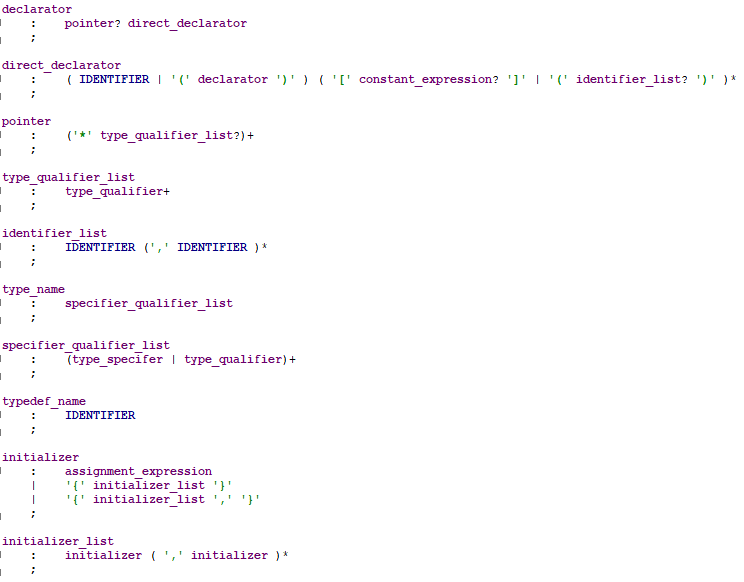
这一部分的产生式包含了C 语言中所有的高层（全局变量以及函数定义）语法。其中， translation\_unit 是最初始的语法单元，表示整个程序。每个 external\_declaration 表示一个全局变量、结构体或函数的定义。 function\_definition 函数的定义由类型描述符、函数声明/头、及函数体组成。

1. 语句部分（Statements），定义如下：

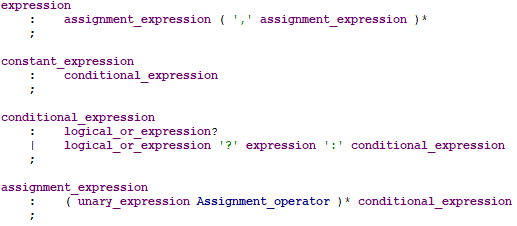
包括标记语句、复合语句、表达语句、选择语句、循环语句与跳转语句。分别定义如下：

其中，标记语句主要运用在switch选择语句中；复合语句是由一对花括号括起来的语句块；选择语句包括if、if……else、switch形式；循环语句包括while、do……while、及两张for循环形式；跳转语句包括goto、continue、break、return等形式。

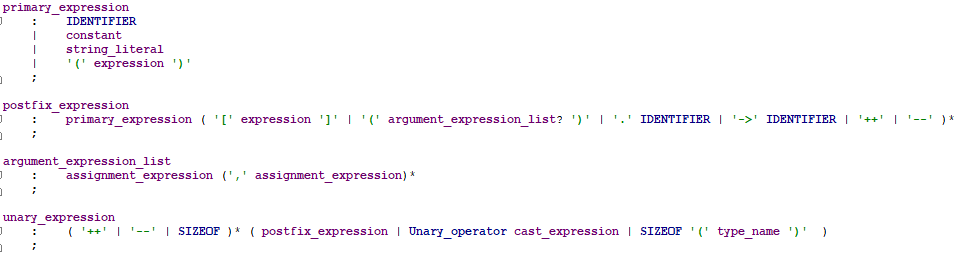
1. 声明部分（Declarations），定义如下：

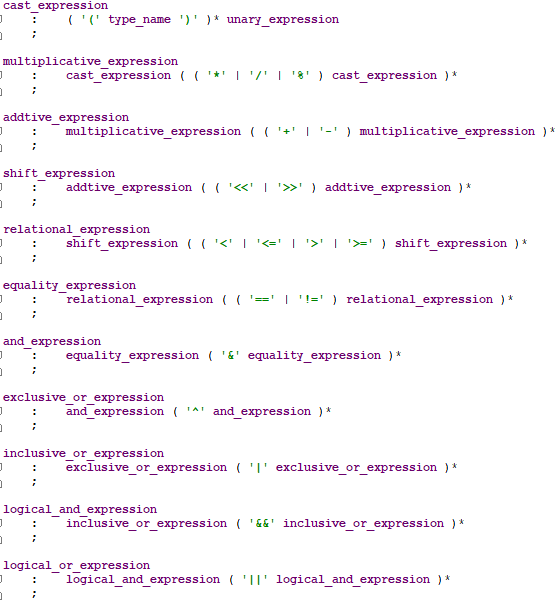
这一部分的产生式主要与变量和函数的定义有关，声明主要由类型符 specifier 和 声明符 declarator 组成。declaration\_specifier 包括储存类型（static、extern……）、变量类型（int、long……）和修饰符（const、volatile）等。

declarator 由指针与直接声明符组成。直接声明符可以为一个标识符或者后跟若干括号的数组。

1. 表达式部分（Expressions），定义如下：

这一部分的产生式主要与表达式有关：表达式可以演化出的形式多种多样，但总体上可分为下面几种：

1. 包含二元运算符的表达式：赋值表达式（assignment\_expression）、逻辑与（logical\_and\_expression）、逻辑或（logical\_or\_expression）、关系表达式（relational\_expression）以及四则运算表达式（+-\*/%等）。
2. 包含一元运算符的表达式（unary\_expression）：括号表达式、取负以及逻辑非等。
3. 不包含运算符但又比较特殊的表达式（postfix\_expression）：函数调用表达式、数组访问表达式以及结构体访问表达式等等。
4. 最基本的表达式（primary\_expression）：整型常数、浮点型常数以及普通变量等。

由于表达式中可以包含各种各样的运算符，为了消除潜在的二义性问题，需要区分运算符的优先级（precedence）以及结合性（associativity），最终形成以下类似递归的文法序列。

最终使用设计的文法，作出少许修改与简化后，识别自己编写的测试程序：

int fun(int a,int b){

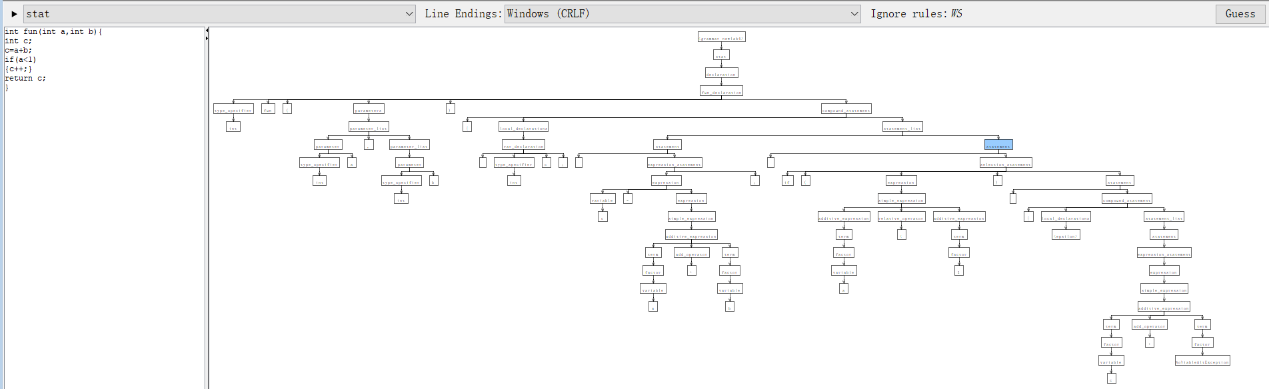
int c;

c=a+b;

if(a<1)

{c++;}

return c;

}

语法树结果如下：

1. **实验心得体会**

本次实验是编译原理与设计的第四次实验，本次实验，让我进一步认识到了语法分析过程，并有机会仔细阅读了C语言的规范文件。对C语言的产生过程和前端的编译过程有了更深一步的认识，了解程序设计语言的演化过程和相关标准的制定过程，加深掌握文法的概念，并能够在一定程度内，使用文法对指定的语言进行描述。

同时这次实验也进一步了加深我对C语言语法的理解，了解了更多不常使用的语句或者过程。如此繁重细致的C语言的文法规则也体现了C语言作为一门经久不息语言的内在逻辑。

需要注意的是，Antlr的语法较为特殊，同时编写Antlr文档时需注意消除文法的左递归与二义性等诸多问题，最终才能呈现出一个rubust的语法分析程序。