文法设计实验报告

**一、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。

**二、实验内容**

本次实验需要依次完成以下三项内容：

\* 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。

\* 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 。

\* 根据自己定义的文法子集，推导出"Hello Word"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。

**三、实验过程**

**3.1 对语言规范的理解**

广义上，语言规范是指使用某种语言的人所应共同遵守的语音、词汇、语法、书写等方面的标准和典范。机器使用的机器语言有其语言规范，人们日常使用的自然语言也有其语言规范。当出现语法或语义上的歧义时，最好的方法是制定新的语法规范来避免这种二义性的产生，人类往往也会根据不同的环境、语境对其做出不同的理解和判断。然而机器对于语言的理解就只有语言本身，所以我们需要制定严格的语法规范，使机器在进行机器语言的识别时做出唯一的判断，所以程序设计语言具有语法严格、结构正规、便于计算机处理等特点。

语言由语法和语义两部分构成。在C语言的规范中，对于语法的定义大体分为以下四类：表达式、声明、语句和定义，另外还包括由多个组成的一些序列等。而语义分析主要是针对内容的检查，检查代码运行的合法性，例如类型推导检查、数组索引检查、函数实现检查等。

**3.2 C语言文法子集描述**

在文法定义中会使用到以下词法：

标识符identifier, 基本数据类型关键字type\_specifier，常量constant，字符串字面量string

3.2.1 函数定义

可以简单地认为C语言程序主体由函数定义构成，将其文法定义如下：

<program>→<function\_definition>|<function\_definition ><program>

<function\_definition>→<type\_specifier><identifier>’(’’)’<statement>

<parameter\_definition>→<declarator>|<declarator>’,’<parameter\_definition>

3.2.2 表达式

C语言表达式文法可以简单定义如下：

<expression>→<primary\_expression>

|<expression><postfix\_expression>

|<unary\_operator><expression>

|<expression><binary\_operator><expression>

考虑到运算符的优先级，按照运算符优先级递增将表达式文法重新定义如下：

\* 赋值表达式

<expression>→<assignment\_expression>

<assignment\_expression>→<logic\_or\_expression>

|<unary\_expression><assignment\_operator><assignment\_expression>

<assignment\_operator>→'=' | '\*=' | '/=' | '%=' | '+=' | '-=' |

\* 逻辑或表达式

<logic\_or\_expression>→<logic\_and\_expression>

|<logic\_and\_expression>’||’<logic\_or\_expression>

\* 逻辑与表达式

<logic\_and\_expression>→<equality\_expression>

|<equality\_expression>’&&’ <logic\_and\_expression>

\* 等值表达式

<equality\_expression>→<relation\_expression>

|<realtion\_expression><equality\_operator><equality\_expression>

< equality\_operator >→’==’|’!=’

\* 关系运算表达式

<relation\_expression>→<additive\_expression>

|<additive\_expression><realtion\_operator><relation\_expression>

<realtion\_operator>→’>’|’<’|’>=’|’<=’

\* 加减运算表达式

<additive\_expression>→<multiplicative\_expression>

|<multiplicative\_expression><additive\_operator><additive\_expression>

<additive\_operator>→’+’|’-‘

\* 乘除运算表达式

<multiplicative\_expression>→<unary\_expression>

|<unary\_expression>< multiplicative \_operator>< multiplicative \_expression>

<multiplicative \_operator>→’\*’|’/‘|’%’

\* 一元运算表达式

<unary\_expression>→<postfix\_expression>

|<unary\_operator><unary\_expression>

<unary\_operator>→’++’|’--‘|’+’ |’-’ |’!’ |’\*’ |’&’

\* 后缀运算表达式

<postfix\_expression>→<primary\_expression>

|<postfix\_expression><postfix\_operator>

<postfix\_operator>→’++’|’--‘

\* 基本表达式

<primary\_expression>→<identifier>|<constant>|<string>|’(‘<expression>’)’

以上表达式覆盖了大部分常用运算符和操作符，但位运算和相关赋值运算符等没有考虑在内。

最后加入分号作为表达式语句的终结符：

<expression\_statement>→<expression>’;’

3.2.3 变量声明

仅考虑了基本变量的声明。

<declarator>→<type\_specifier><identifier>|<declarator>’=’<assignment\_expression>

<declarator\_statement>→<declarator>’;’

3.2.4 语句

总体上语句的文法定义如下：

<statement>→’{‘<compound\_statement>’}’

|<expression\_statement>

|<selection\_statement>

|<loop\_statement>

|<declarator\_statement>

<compound\_statement>→<statement>|<statement><compound\_statement>

3.2.5 分支语句

仅考虑了if-else语句，未考虑包含else if和switch-case的语句。

<selection\_statement>→’if’’(‘<expression>’)’<statement>

|’if’’(‘<expression>’)’<statement>’else’<statement>

3.2.6 循环语句

<selection\_statement>→’while’’(‘<expression>’)’<statement>

|’do’<statement>’while’’(‘<expression>’)’

|’for’’(‘<expression>’;’<expression>’;’<expression>’;’’)’

**3.3 程序推导过程**

推导的最简程序如下：

int main() {

    int x=0;

    while(x < 5)

        x++;

    if (x<10)

        x = 10;

}

推导出的语法树如下：

<program>

    <function\_definition>

        <type\_specifier>

            int

        <identifier>

            main

        (

        )

        <statement>

            {

            <compound\_statement>

                <statement>

                    <declarator\_statement>

                        <declarator>

                            <type\_specifier>

                                int

                            <identifier>

                                x

                            =

                            <assignment\_expression>

                                <logic\_or\_expression>

                                    <logic\_and\_expression>

                                        <equality\_expression>

                                            <relation\_expression>

                                                <additive\_expression>

                                                    <multiplicative\_expression>

                                                        <unary\_expression>

                                                            <postfix\_expression>

                                                                <primary\_expression>

                                                                    <constant>

                                                                        0

                            ;

                <statement>

                    <loop\_statement>

                        while

                        (

                        <expression>

                            <assignment\_expression>

                                <logic\_or\_expression>

                                    <logic\_and\_expression>

                                        <equality\_expression>

                                            <relation\_expression>

                                                <additive\_expression>

                                                    <multiplicative\_expression>

                                                        <unary\_expression>

                                                            <postfix\_expression>

                                                                <primary\_expression>

                                                                    <identifier>

                                                                        x

                                                <relation\_operator>

                                                    <

                                                <additive\_expression>

                                                    <multiplicative\_expression>

                                                        <unary\_expression>

                                                            <postfix\_expression>

                                                                <primary\_expression>

                                                                    <constant>

                                                                        5

                        )

                        <statement>

                            <expression\_statement>

                                <expression>

                                    <assignment\_expression>

                                        <logic\_or\_expression>

                                            <logic\_and\_expression>

                                                <equality\_expression>

                                                    <relation\_expression>

                                                        <additive\_expression>

                                                            <multiplicative\_expression>

                                                                <unary\_expression>

                                                                    <postfix\_expression>

                                                                        <primary\_expression>

                                                                            <identifier>

                                                                                x

                                                                        <postfix\_operator>

                                                                            ++

                                ;

                <statement>

                    <selection\_statement>

                        if

                        (

                        <expression>

                            <assignment\_expression>

                                <logic\_or\_expression>

                                    <logic\_and\_expression>

                                        <equality\_expression>

                                            <relation\_expression>

                                                <additive\_expression>

                                                    <multiplicative\_expression>

                                                        <unary\_expression>

                                                            <postfix\_expression>

                                                                <primary\_expression>

                                                                    <identifier>

                                                                        x

                                                <relation\_operator>

                                                    <

                                                <additive\_expression>

                                                    <multiplicative\_expression>

                                                        <unary\_expression>

                                                            <postfix\_expression>

                                                                <primary\_expression>

                                                                    <constant>

                                                                        10

                        )

                        <statement>

                            <expression\_statement>

                                <expression>

                                    <assignment\_expression>

                                        <unary\_expression>

                                            <postfix\_expression>

                                                <primary\_expression>

                                                    <identifier>

                                                        x

                                        <assignment\_operator>

                                            =

                                        <assignment\_expression>

                                            <logic\_or\_expression>

                                                <logic\_and\_expression>

                                                    <equality\_expression>

                                                        <relation\_expression>

                                                            <additive\_expression>

                                                                <multiplicative\_expression>

                                                                    <unary\_expression>

                                                                        <postfix\_expression>

                                                                            <primary\_expression>

                                                                                <constant>

                                                                                    10

                                ;

            }