编译原理与设计实验报告

姓名：卜梦煜　　学号：1120192419　　班级：07111905

1. 实验名称

文法设计与验证实验

2. 实验目的

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。

3. 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。

4. 实验环境

Intellij IDEA Community 2021.3.2 + 插件 ANTLR v4 1.17

5. 实验过程与步骤

本次实验阅读了 C 语言和 Java 语言的规范草稿，参考 C11 语言规范划分 C 语言子集，并完成子集内语言的文法设计任务，最终利用 IDEA 的 ANTLR v4 插件完成语法验证。

5.1 理解语言规范

语言标准规范是每个语言的说明文档，定义了语言的字符集、词法规则、语法规则和语义规则，也包括了对程序结构、编译过程、标准库程序以及语言实现等各方面的内容。一个标准规范描述为不同视角的参与者提供了一个一致的参考标准，例如程序员可以根据规范标

准编写程序，系统设计者则可以根据这一标准规范设计编译器、完成运行时环境和库的构造和实现。由于编译器基于同一标准规范设计，因此才能对程序员编写完成的各式各样的程序进行不同层面验证，并将其最终翻译为目标代码。同理，所有的程序设计人员都基于同一个标准设计程序，按照给定的接口调用库函数，因此大家才能交换共享并理解对方的代码，并协同工作完成大规模软件项目的开发。

（1）C11-C 语言规范

该语言规范规定了 C 语言的字符集（5.2 节）、语言的词法规则（6.4 节）、语言的语法规则（6.5-6.11 节）、库等内容。其中字符集包括基本字符集与扩展字符集；词法规则规定了 5 种 token：keyword、identifier、constant、string-literal、punctuation，包括了 C 程序中使用的词法元素；语法规则包括表达式语句、声明语句、表达式与块等的生成规则。

语言规范的意义在于，对 C 语言源码输入、到单词识别、到语法成分解析、并最终到编译成可执行程序的编译全过程进行了严格的说明规范，保证了语言的严谨性，尽可能消除二义性。

（2）Java SE12 语言规范

Java 语言规范的内容与 C11 相似，并针对 Java 语言的一些高级特性作了规定，如包、接口、模块、线程与锁等，更好地支持了 Java 语言面向对象、多线程等特性。

5.2 选定 C 语言子集并用文法描述

（1）划分 C 语言子集

本实验目标是支持 C 语言的基本运算操作，据此划定 C 语言简单子集如下：

C语言简单子集：

- 数据类型：简单数据类型(int float double char string)，不支持(指针 数组 enum union struct)
- 数据操作：赋值(= *= /= %= += -= <<= >>= &= ^= |=)、算术运算(+ - * / %)、逻辑运算(&& || !)、位运算(& | ^ << >>)、关系运算(< > == <= >= !=)、三目运算(?:)，自定义函数，支持运算符优先级(参考C11标准)，不支持单目运算(++ -- [] . -> &等)、类型转换
- 程序结构：循环语句(while for)、分支语句(if else)
- 关键字：int float double char main while for if else
- 符号集合：( ) { } ; 数据操作中定义的符号
- 部分不支持的内容：宏、头文件、注释
- 其他内容：
  - 表达式：与C11定义的标准一致
  - 赋值语句：支持"简单变量 赋值符号 赋值语句"的格式，不支持左值存在数组、自增自减、指针等情况，支持的赋值符号为C11定义的赋值符号
  - 循环语句：支持while for循环结构，不支持do-while循环结构
  - 分支语句：支持if、if-else、if-else嵌套的分支结构，不支持switch分支结构

（2）字母表

使用的字母表参考 C11 中规定的字母表，并针对划定的 C 语言子集做了调整，包括大小写字母 52 个，数字 10 个，以及其他符号 22 个。

```
Letters:    // 字母
    [a-zA-Z]
Digits:    // 数字
    [0-9]
GraphicCharacters:    // 22个其他字符
    [!"%'()*+,-/:;<=>?^_{|}]
```

（3）词法规则设计

该部分使用 ANTLR 文法描述方法，词法规则的含义与词法规则实验类似，识别变量类型、关键词、标识符、常量、字符串字面量，lexer 文件关键部分定义如下：

```
// 类型名
TypeSpecifier: 'void' | 'int' | 'float' | 'double' | 'char' | 'string' | 'signed' | 'unsigned';

// 关键词
MAIN: 'main';
WHILE: 'while';
FOR: 'for';
IF: 'if';
ELSE: 'else';
CONTINUE: 'continue';
BREAK: 'break';
RETURN: 'return';

// 标识符
Identifier: NonDigit (NonDigit | Digit)*;

// 常量
Constant : IntegerConstant | FloatConstant | CharConstant;

// 字符串字面量
StringLiteral : ('u8' | 'u' | 'U' | 'L')? '"' (Char | EscapeChar | OctalEscapeChar | HexadecimalEscapeChar)+ '"';
```

（4）语法规则设计

该部分使用 ANTLR 文法描述方法，参考 C11 标准，对划定的 C 语言子集的语法进行设计。

文法分析入口如下：

```
// 入口
translationUnit:
    externalDeclaration
    | translationUnit externalDeclaration
    ;

externalDeclaration:
    functionDefinition
    | declaration expression ';'
    ;

functionDefinition:
    declaration (Identifier | MAIN) '(' declarationList? ')' block
    ;

declarationList:
    declaration
    | declarationList declaration
    ;

block:
    statement block?
    | '{' statement block? '}'
    ;

statement:
    declaration? expression ';'
    | selectionStatement
    | iterationStatement
    | jumpStatement ';'
    ;
```

表达式 Expression 定义如下:

```
// 表达式
expression:
    assignmentExpression
    | expression ',' assignmentExpression
    ;
```

赋值语句 assignmentExpression 包括条件表达式 conditionalExpression 与“标识符+赋值运算符+赋值语句”格式的一般赋值语句, 条件表达式为支持运算符优先级, 嵌套结构较复杂; 一般赋值语句为常用赋值语句，支持连续赋值。定义如下:

```
// 赋值语句
assignmentExpression:
    conditionalExpression
    | Identifier assignmentOperator assignmentExpression
    ;

// 运算符优先级
conditionalExpression:
    logicalOrExpression
    | logicalOrExpression '?' expression ':' conditionalExpression
    ;

logicalOrExpression:
    logicalAndExpression
    | logicalOrExpression '||' logicalAndExpression
    ;

logicalAndExpression:
    inclusiveOrExpression
    | logicalAndExpression '&&' inclusiveOrExpression
    ;

inclusiveOrExpression:
    exclusiveOrExpression
    | inclusiveOrExpression '|' exclusiveOrExpression
    ;

exclusiveOrExpression:
    andExpression
    | exclusiveOrExpression '^' andExpression
    ;

andExpression:
    equalityExpression
    | andExpression '&' equalityExpression
    ;
```

```
equalityExpression:
    relationalExpression
    | equalityExpression '==' relationalExpression
    | equalityExpression '!=' relationalExpression
    ;

relationalExpression:
    shiftExpression
    | relationalExpression '<' shiftExpression
    | relationalExpression '>' shiftExpression
    | relationalExpression '<=' shiftExpression
    | relationalExpression '>=' shiftExpression
    ;

shiftExpression:
    additiveExpression
    | shiftExpression '<<' additiveExpression
    | shiftExpression '>>' additiveExpression
    ;

additiveExpression:
    multiplicativeExpression
    | additiveExpression '+' multiplicativeExpression
    | additiveExpression '-' multiplicativeExpression
    ;

multiplicativeExpression:
    primaryExpression
    | multiplicativeExpression '*' primaryExpression
    | multiplicativeExpression '/' primaryExpression
    | multiplicativeExpression '%' primaryExpression
    ;

primaryExpression:
    Identifier
    | Constant
    | StringLiteral
    | '(' expression ')'
    ;
```

```
assignmentOperator:
    '=' | '*=' | '/=' | '%=' | '+=' | '-=' | '<<=' | '>>=' | '&=' | '^=' | '|='
    ;
```

分支语句 selectionStatement 支持 if、if-else、if-else 多重嵌套结构的条件分支语句，定义如下：

```
// 分支语句
selectionStatement:
    IF '(' expression ')' block
    | IF '(' expression ')' block ELSE block
    ;
```

循环语句 iterationStatement 支持 while、for 循环，不支持 do-while 循环，定义如下：

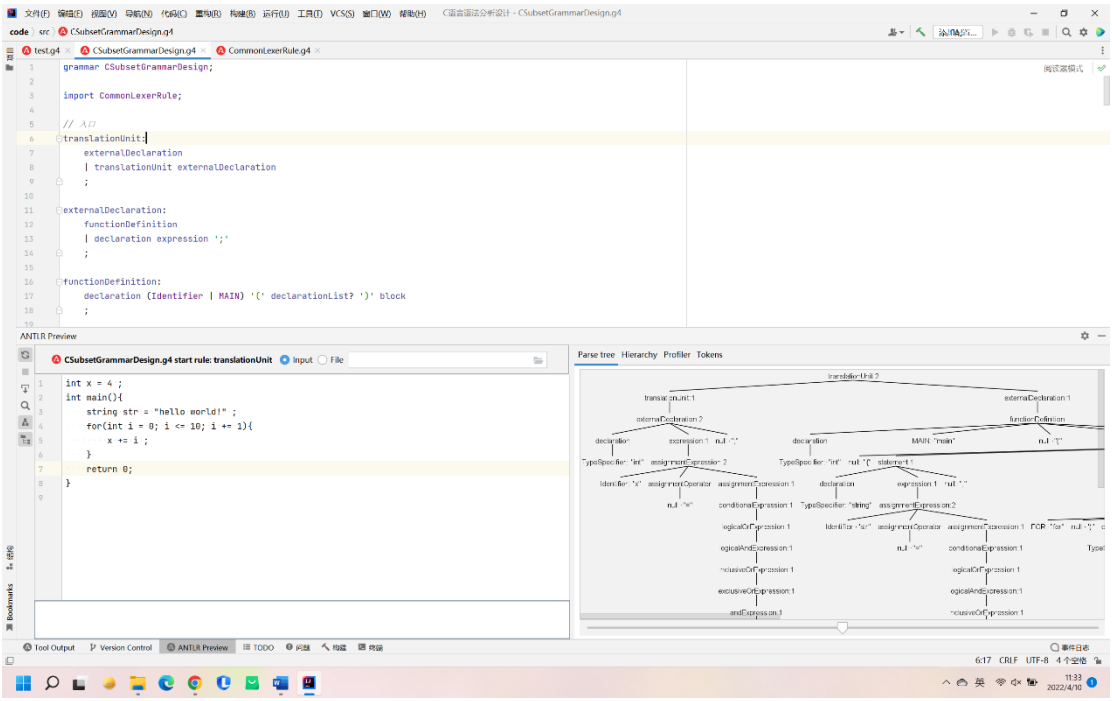
```
// 循环语句
iterationStatement:
    WHILE '(' expression ')' block
    | FOR '(' (declaration? expression)? ';' expression? ';' expression? ')' block
    | FOR '(' (declaration expression)? ';' expression? ')' block
    ;
```

5.3 设计测试程序验证文法

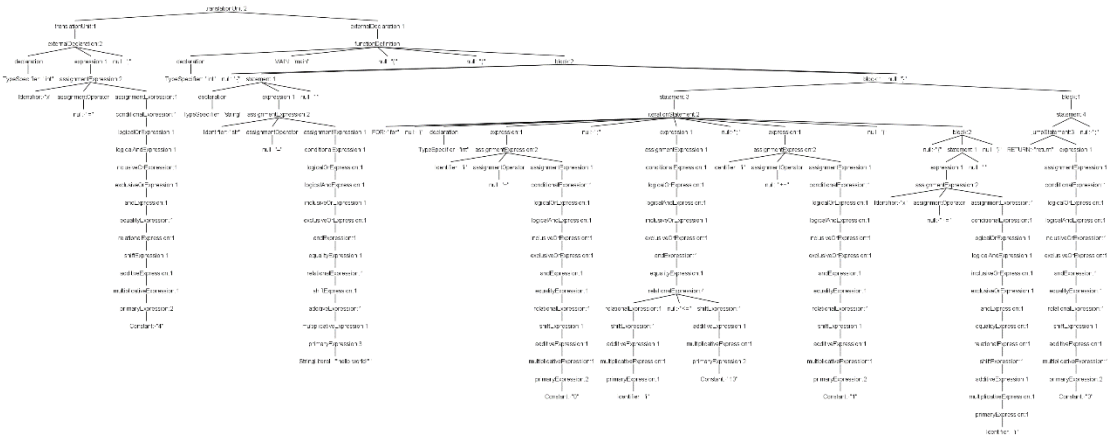
设计的 C 语言程序如下：

```
int x = 4 ;
int main(){
    string str = "hello world!" ;
    for(int i = 0; i <= 10; i += 1){
        x += i ;
    }
    return 0;
}
```

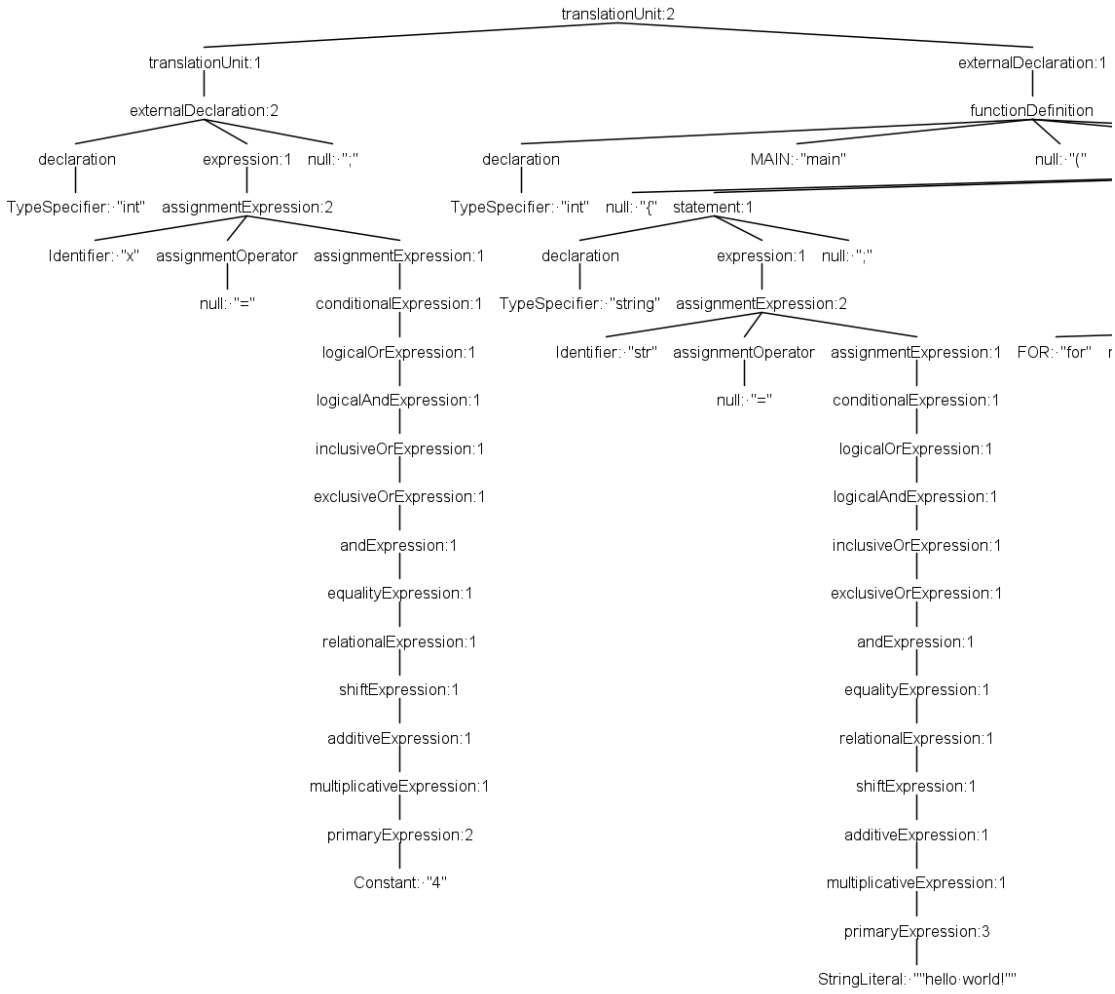
运行截图如下：

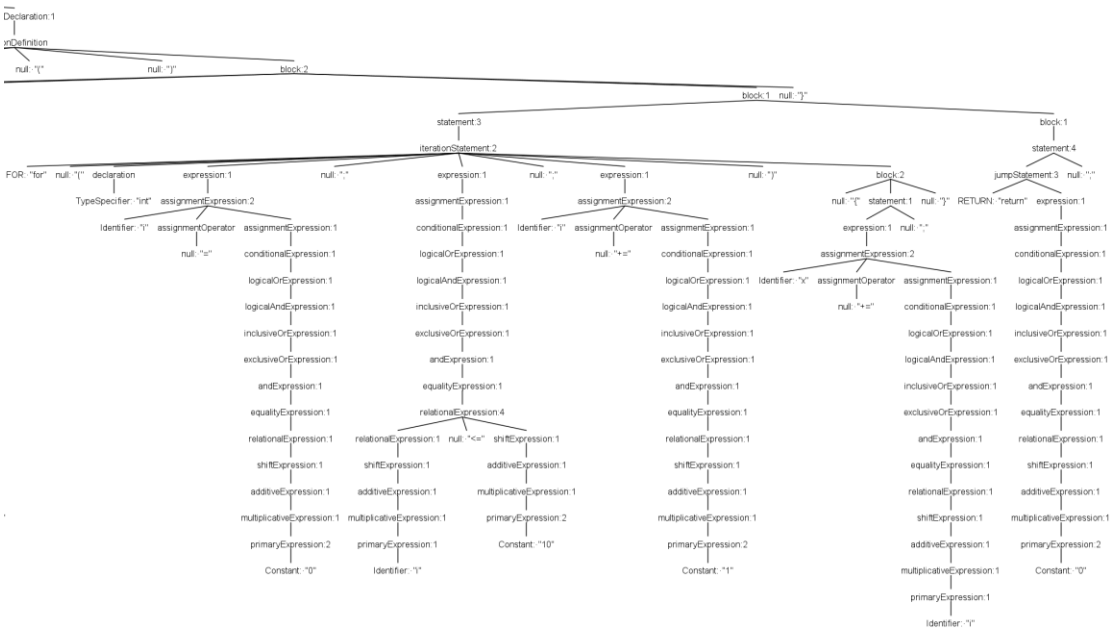


生成的语法树结构如下:



部分结构如下:


translationUnit:2
translationUnit:1
externalDeclaration:1
externalDeclaration:2
functionDefinition
declaration
expression:1
null:·";"
declaration
MAIN:·"main"
null:·"("
TypeSpecifier:·"int"
assignmentExpression:2
TypeSpecifier:·"int"
null:·"{"
statement:1
Identifier:·"x"
assignmentOperator
assignmentExpression:1
declaration
expression:1
null:·";"
null:·"="
conditionalExpression:1
TypeSpecifier:·"string"
assignmentExpression:2
logicalOrExpression:1
Identifier:·"str"
assignmentOperator
assignmentExpression:1
FOR:·"for"
logicalAndExpression:1
null:·"="
conditionalExpression:1
inclusiveOrExpression:1
logicalOrExpression:1
exclusiveOrExpression:1
logicalAndExpression:1
andExpression:1
inclusiveOrExpression:1
equalityExpression:1
exclusiveOrExpression:1
relationalExpression:1
andExpression:1
shiftExpression:1
equalityExpression:1
additiveExpression:1
relationalExpression:1
multiplicativeExpression:1
shiftExpression:1
primaryExpression:2
additiveExpression:1
Constant:·"4"
multiplicativeExpression:1
primaryExpression:3
StringLiteral:·""hello·world!""


Declaration:1
nDefinition
null:·"{"
null:·"}"
block:2
block:1
null:·"}"
statement:3
block:1
iterationStatement:2
statement:4
FOR:·"for"
null:·"("
declaration
expression:1
null:·";"
expression:1
null:·";"
expression:1
null:·")"
block:2
jumpStatement:3
null:·";"
TypeSpecifier:·"int"
assignmentExpression:2
assignmentExpression:1
assignmentExpression:2
null:·"{"
statement:1
null:·"}"
RETURN:·"return"
expression:1
Identifier:·"i"
assignmentOperator
assignmentExpression:1
Identifier:·"i"
assignmentOperator
assignmentExpression:1
expression:1
null:·";"
null:·"="
conditionalExpression:1
null:·"+="
conditionalExpression:1
assignmentExpression:2
conditionalExpression:1
logicalOrExpression:1
logicalOrExpression:1
Identifier:·"x"
assignmentOperator
assignmentExpression:1
logicalAndExpression:1
logicalAndExpression:1
null:·"+="
conditionalExpression:1
inclusiveOrExpression:1
inclusiveOrExpression:1
logicalOrExpression:1
exclusiveOrExpression:1
exclusiveOrExpression:1
logicalAndExpression:1
andExpression:1
andExpression:1
inclusiveOrExpression:1
equalityExpression:1
equalityExpression:1
exclusiveOrExpression:1
relationalExpression:4
andExpression:1
relationalExpression:1
relationalExpression:1
shiftExpression:1
equalityExpression:1
relationalExpression:1
null:·"<="
shiftExpression:1
shiftExpression:1
additiveExpression:1
relationalExpression:1
shiftExpression:1
additiveExpression:1
additiveExpression:1
additiveExpression:1
shiftExpression:1
multiplicativeExpression:1
multiplicativeExpression:1
multiplicativeExpression:1
additiveExpression:1
primaryExpression:2
primaryExpression:1
multiplicativeExpression:1
primaryExpression:2
primaryExpression:1
Constant:·"0"
Identifier:·"i"
Constant:·"10"
Constant:·"1"
Identifier:·"i"
Constant:·"0"

6. 实验心得体会

这次语法分析与验证实验中我有如下收获：

（1）阅读了 C11 规范文件，对 C 语言代码的生成过程有了更深一步的认识，理解了语言规范的作用、初步掌握了 C11 标准下 C 语言的文法生成规则，并能够对划分的 C 语言子集用文法进行描述。

（2）深刻感受到 C 语言设计的严谨性，进一步加深了对 C 语言语法的理解。

（3）学习使用 ANTLR 工具，利用该工具对自定义的 C 语言文法实时测试，方便直观地验证所写文法的正确性。