实验一报告

171860513 刘天宇　171860573 宾向荣

一、实验目标

使用 GNU Flex/Bison 辅助词法和语法分析，编写一个程序对使用 c—语言书写的源代码进行词法和语法分析，并打印分析结果。其中分析结果包括错误分析以及正确情况下的语法树。

二、实验过程

1、词法分析

（1）模块功能

将输入文件中的字符流组织成为词法单元流，并且在某些字符不符合程序设计语言词法规范时报告相应的错误

（2）步骤描述

编写 flex 源代码，在规则部分编写正则表达式和相应的响应函数。

2、语法分析

（1）模块功能

读入词法单元流，并在匹配规范的情况下构建起输入程序的静态结构。

（2）步骤描述（简）

① 为生成树的节点定义了一个类型，用于描述生成树的每个节点的信息。

```
struct Node{
    int flag;//current type, suc
    //nonterminal 0, int 1, floa
    int line;//now line
    char type[16]; //morpheme
    char text[33]; //ID
    struct Node *left;
    struct Node *right;
};
```

② 在.y 文件中的规则部分书写正确的产生式，并且定义相应的行为。

③ 每条产生式的归约，都生成一棵子树，直到最后归约到 program，便生成了最后的生成树。

④ 归约结束后将生成树输出。

（3）细节分析：

① 选择了二叉树来存储生成树，这样节省了程序所需空间，并且在输出的时候只需要中序遍历二叉树即可输出生成树。

② 在语法树中，每个非终结符都需要输出行号，起初，我直接使用 yylineno，但是有些产生式跨越多行便会导致错误，故需要把产生式右端的符号中行号最小的赋值给产生式右端的行号。由于产生式的代码跨越多行，因此期间的终结符的行号的获取也会产生错误，但是如果我们直接采用 bison 内置的相关的宏 YY_USER_ACTION，便可以直接获取终结符的正确的位置。

③ 处理正确的注释：当读入“\\”和“*”时利用 input 把被注释掉的内容读入，但是不做处理也不返回词素。

④ 语法树中的缩进是一个不容易解决的问题，但是由于我使用了二叉树来存储语法树，我只需要设置一个变量并初始化为 0，在中序遍历的时候每次进入左儿子都将该变量加 1，进入右儿子时不变即可，这是因为在二叉树存储多叉树时，左儿子意味着层数加 1。以下是打印语法树的语句。

```
printTree(head->left, counts + 1);
printTree(head->right, counts);
```

3、错误检测

（1）模块功能

实现对待编译代码中的词法、语法错误进行错误类型分析、错误位置报告与**错误内容描述**。

（2）成果展示

构造了具有代表性的测试用例（下图为选段），展示本代码语法分析错误检测和恢复的**鲁棒性**。

```
22  int arr6 [2]5];
23  int arr7 2][5];
24  int arr8 [2][5;
25  int arr9 [2,5];
26  int a, arr10[2];
27  int arr11[2, arr12 [2], arr13 3];
28  int [2];
29
30  // Global Definitions (to be editted)
31  int main1() { !a; }
32  int k m,n;
33  int j;
34  int  {}
35  s,ss;
36  INT s,ss;
37  int s ss;
38  egd;
39  func0(int x) {}
40  int l;
41  // INT func2(int y) { return 1; } // apt to conflict with << error . ExtDefList >>
42
43  int j,k;
44
45  // Function Definition
46  int func1(int x, z) { return 1; }
47  int func2(int x,) { return 2; }
48  int func3(int x, INT y) { return 3; }
49  int func4(int z) { int def1 = 1; k = 1; int def2 = 1; }
50  int func5(int z) { +++ }
51  int func6(int m) { ; } // empty stmt not allowed
52  int func7(int m) { int s ss; } // a Stmt-level problem, handled already
53
54  // Statements
55  int func8(int z) { return int; }
56  int func9(int a) { if a > b) a = b; }
57  int func10(int b) { if () a = b; }
58  int func11(int c) { if (a > b a = b + a; }
59  int func9_2(int a) { if a > b) return a; else return b; }
60  int func10_2(int b) { if () return a; else return b; }
61  int func11_2(int c) { if (a > b return a; else return b; }
62  int func_given() { if (a > b) a = b else return b; }
```

```
Error type B at Line 36: Missing or Invalid separator ','
Error type B at Line 36: Missing or Invalid specifier
Error type B at Line 37: Missing or Invalid separator ','
Error type B at Line 38: Missing or Invalid specifier
Error type B at Line 39: Missing or Invalid function return type
Error type B at Line 46: Invalid function declarator
Error type B at Line 47: Invalid function declarator
Error type B at Line 48: Invalid function declarator
Error type B at Line 49: Invalid statement
Error type B at Line 49: Invalid complex statements
Error type B at Line 50: Invalid complex statements
Error type B at Line 51: Invalid complex statements
Error type B at Line 52: Missing or Invalid assignment operator '=' or separator ','
Error type B at Line 55: Invalid return statement
Error type B at Line 56: Invalid IF statement: Missing or Invalid lbrace '('
Error type B at Line 57: Invalid IF statement: Missing or Invalid if-condition
Error type B at Line 58: Invalid IF statement: Missing or Invalid rbrace ')'
Error type B at Line 59: Invalid IF-ELSE statement: Missing or Invalid lbrace '('
Error type B at Line 60: Invalid IF-ELSE statement: Missing or Invalid if-condition
Error type B at Line 61: Invalid IF-ELSE statement: Missing or Invalid rbrace ')'
Error type B at Line 62: Invalid statement with Incomplete expressions
Error type B at Line 63: Invalid WHILE statement: Missing or Invalid lbrace '('
Error type B at Line 64: Invalid WHILE statement: Missing or Invalid while-condition
Error type B at Line 65: Invalid WHILE statement: Missing or Invalid rbrace ')'
Error type B at Line 69: Invalid definition with incorrect identifier
Error type B at Line 69: Invalid definition with incorrect identifier
Error type B at Line 70: Missing or Invalid assignment operator '=' or separator ','
Error type B at Line 71: Missing or Invalid assignment operator '=' or separator ','
Error type B at Line 72: Invalid definition with incorrect identifier
Error type B at Line 76: Invalid expression inside braces
Error type B at Line 77: Invalid IF statement: Missing or Invalid if-condition
Error type B at Line 78: Invalid statement with Incomplete expressions
Error type B at Line 80: Invalid expression: check whether there's a missing rbrace ')' ?
Error type B at Line 82: Invalid expression inside braces
Error type B at Line 83: Invalid statement with Incomplete expressions
Error type B at Line 84: Invalid return statement
Error type B at Line 87: Invalid argument passing
Error type B at Line 88: Missing or Invalid separator ',' in argument passing
Error type B at Line 88: Missing or Invalid separator ',' in argument passing
Error type B at Line 90: Invalid statement with Incomplete expressions
Error type B at Line 92: Invalid expression in braces '[]'
Error type B at Line 92: Invalid expression in braces '[]'
Error type B at Line 93: Invalid statement with Incomplete expressions
Error type B at Line 94: Invalid expression: check whether there's a missing rbrace ']' ?
```

（3）步骤描述

① 词法模块错误检测

在 lexical.l 规则部分，增加词法错误类型条目，并辅以下述处理过程。

I. 对于不符合定义的标识符，报错“Invalid Identifier”，同时执行“return ID;”以便利语法分析。

II. 对于不在词法定义内的字符，报错“Mysterious character”。

III. （选做）对于多行注释“/**/”不能正确匹配的情况，其类型为**语法错误**，但在词法模块中给出。若匹配到“/*”，则不停读入字符至匹配到第一个“*/”；若匹配到“*/”，则报错“No match for ‘*/’”。

② 语法模块错误检测

本代码为各种类型的语法错误提供了三**十余种**检错机制，体现为 syntax.y 中用宏 ErrReport(str, lin) 报告规约到 error 产生式的情况，并保证无二义性。检错规则如下：

error 产生式一览表

error 产生式	**报错内容**	**解释**
I. 全局定义和声明		
ExtDefList → **error** ExtDefList	Invalid global definition	高层定义直接出错
ExtDef → **error** FunDec CompSt	Missing or Invalid function return type	函数返回类型错误
ExtDef → **error** ExtDefList SEMI	Missing or Invalid specifier	变量类型说明符错误
ExtDef → Specifer FunDec **error** SEMI	Missing or Invalid function body	C--中不允许函数声明
ExtDef → Specifier **error** CompSt	Missing or Invalid function declarator	函数接口定义错误
ExtDecList → VarDec **error** ExtDecList	Missing or Invalid separator ‘,’	函数接口分隔符错误
StructSpecifier → STRUCT **error** DefList RC	Missing or Invalid structure lbrace ‘{‘	结构体成员定义语句左括号位置附近出错

StructSpecifier → STRUCT OptTag LC DefList **error**	Missing or Invalid structure rbrace ' } '	结构体成员定义语句右括号位置附近出错
VarDec → VarDec **error** INT RB	Missing or Invalid array lbrace ' ['	数组定义左括号出错
VarDec → VarDec LB **error** RB	Missing or Invalid array size	数组定义长度声明错
VarDec → VarDec LB INT **error**	Missing or Invalid array rbrace '] '	数组定义右括号出错
VarDec → **error** LB INT RB	Missing or Invalid array name	数组定义标识符出错
FunDec → ID LP **error** RP	Invalid function parameter list	函数定义接口形参错
VarList → COMMA **error** ParamDec	Invalid function declarator	函数参数声明列表错
ParamDec → Specifier **error** VarDec	Invalid paramter declaration	函数参数声明格式错
II. 语句和复合语句		
CompSt → LC **error** RC	Invalid Complex Statements	复合语句内部混乱
StmtList → Stmt **error** StmtList	Invalid Statement	单句或多句语句出错
Stmt → Exp **error** SEMI	Invalid statement with Incomplete exp	语句含不完整表达式
Stmt → RETURN **error** SEMI	Invalid return statement	返回语句内容出错
Stmt → IF **error** Exp RP Stmt [ELSE Exp]	Invalid IF[-ELSE] statement: missing (	IF[-ELSE]语句缺少'('
Stmt → IF LP **error** RP Stmt [ELSE Exp]	Invalid IF[-ELSE] condition expression	IF[-ELSE]条件错误
Stmt → IF LP Exp **error** Stmt [ELSE Exp]	Invalid IF[-ELSE] statement: missing)	IF[-ELSE]语句缺少')'
Stmt → WHILE **error** Exp RP Stmt	Invalid WHILE statement: missing '('	WHILE 语句缺少'('
Stmt → WHILE LP **error** RP Stmt	Invalid WHILE condition expression	WHILE 条件错误
Stmt → WHILE LP Exp **error** Stmt	Invalid WHILE statement: missing ')'	WHILE 语句缺少')'
III. 局部定义和表达式		
Def → Specifier **error** SEMI	Invalid identifier in local definition	局部定义标识符出错
Dec → VarDec **error** Exp	Missing or Invalid symbol ' , ' or ' = '	标识符分隔方式错误
Exp → LP **error** RP	Invalid expression inside braces	括号内表达式不合法
Exp → LP Exp **error**	Invalid expression: check missing ') '	表达式右括号处出错
Exp → ID LP **error** RP	Invalid argument passing	函数调用传参错误
Exp → LB **error** RB	Invalid expression in braces ' [ ] '	数组元素访问下标错
Exp → LB Exp **error**	Invalid expression: check missing '] '	数组右定界符错误
Args → Exp **error** Args	Missing or Invalid separator ' , ' in argument passing	函数实参列表中分隔符错误

三、实验收获

1、通过实际编程，了解了 Flex 和 Bison 工具，理解了词法和语法分析过程、掌握语法树的构建方法。
2、理解了自底向上语法分析过程的移入—规约机制，能够通过观察 LR(0)状态机的状态转移，为潜在的不同类型语法错误确定合适的 error 产生式。