编译原理（1）实验报告

张语婷-181860140-普通班-1051570722@qq.com

一、 实现功能 基础+选做 1.3

1. 词法分析+选做 1.3
   - 正则表达式：书写 INT/FLOAT/ID 等的正则表达式，利用 INT 型定义 FLOAT 小数点前的部分

```
INT 0|[1-9][0-9]*
FLOAT {INT}\.[0-9]+
ID [_a-zA-Z][0-9_a-zA-Z]*
```

   - 处理注释：分为"//"型和"/*…*/"型，"//"型读入到换行符结束，"/*…*/"型读入到第一个" */"结束，没有" */"则输出 error
   - 生成词法单元：使用 struct Node*结构存储词法/语法单元，记录该词法单元相关信息，并且返回

```
#define YYSTYPE struct Node*
```

```
"struct" {
    struct Node* newnode=(struct Node*)malloc(sizeof(struct Node));
    newnode->child=NULL;
    newnode->brother=NULL;
    newnode->linenumber=0;
    strcpy(newnode->name,"STRUCT\0");
    newnode->judge=0;
    yylval=newnode;
    return STRUCT;
    }
```

   - 使用通配符检查词法错误 A：在所有规定词法单元判断完成后，使用通配符检查非法输入
2. 语法分析
   - 建立语法树：产生式右侧的第一个符号链接成为左侧符号的 child 节点，其余右侧符号依次成为其左边符号的 brother 节点

```
VarList:ParamDec COMMA VarList{
    $$=create_Node($1,"VarList\0",@1.first_line);
    $1->brother=$2;
    $2->brother=$3;}
```

   - 由空串生成的父亲节点：对于产生ε的语法单元，仍然新生成一个节点，但标记其 int_number=0，用于打印语法树时不打印该语法单元

```
OptTag: {
    $$=create_Node(NULL,"OptTag\0",0);
    $$->int_number=0;}
```

   - 由空串生成的祖先节点：查询所有产生式，仅 Program:ExtDefList 、ExtDefList:ε满足只由空串生成祖先节点，此时可能出现祖先节点行号记录错误的问题（由于@1.first_line 判断的节点为ε生成），因此进行特殊处理

```
Program:ExtDefList{
    if($1->int_number==0){
        $$=create_Node($1,"Program\0",$1->linenumber);
    }
```

- 错误恢复：使用 SEMI/RC/RB 做为错误恢复的同步符号，同时利用 yylineno 和 yytext 打印错误发生的行号和相关错误文字

```
printf("Error type B at Line %d: syntax error at \"%s\".\n", yylineno, yytext);
```

3. 语法树打印：
   - 打印单个节点：分为词法单元（ID-TYPE-INT-FLOAT-other）和语法单元（ε生成-非ε生成）
   - 打印树：通过数组以及(int)number 模拟 C++中的栈，head 节点进栈。
     循环：对栈顶节点进行操作：（类似于树的先序遍历）
     1）child !=NULL，则将该 child 节点进栈，depth++；
     2）child =NULL，brother !=NULL，则将栈顶节点替换为其 brother 节点；
     3）child =NULL，brother =NULL，则将栈顶节点 pop()，depth--；

```
if(tree_sort[number-1]->child!=NULL){…
}
else if(tree_sort[number-1]->brother!=NULL){…
}
else{…
}
```

二、 编译语句

make
./parser name.cmm （name.cmm 为 cmm 文件名字）