实验报告

课程：编译原理

211220169 祝明浩 211220182 刘钊瑜

专业：计算机科学与技术

一．功能说明

完成实验必做部分内容和选做 1.2 内容，可以识别科学计数法的词法单元并对错误的科学计数法内容识别为相应的词法错误。

代码结构:

main.c:获取文件输入，同时初始化错误标志以及根节点

node.h:定义了多叉树中树的节点的结构

lexical.l:匹配了词法单元，进行词法错误处理并且将对应的词法单元初始化为树节点

syntax.y：匹配产生式并利用产生式产生语法树和进行错误恢复。

二．编译过程

先使用 bison 对 syntax.y 编译：bison -d syntax.y

再对 lexical.l 编译：flex lexical.l

最后得到可执行文件：gcc main.c syntax.tab.c -lfl -ly -o parser

或：直接采用程序自带的 Makefile 进行编译。

三．亮点介绍

1.科学计数法错误识别：

可以识别形似 1.00e,1.00e+,1.00ee1 这类词法错误，其余格式均报为语法错误。

```
{block} {;}
((([0-9]*\.[0-9]+)|([0-9]+\.))[eE][+-]?)|((([0-9]*\.[0-9]+)|([0-9]+\.))[eE][+-]?{letter}[a-zA-Z0-9]*) {printf("Error type A at line %d:
. {printf("Error type A at line %d: Mysterious characters \'%s\'\n",line,yytext);flexflag=1;}
```

2.语法树节点结构以及相关函数:

树节点结构:

用 sline 来记录每个节点在源文件中出现的行数，并用 stype 记录该节点所代表的属性，并且根据属性的不同，节点中记录对应属性的不同具体数值。

```
typedef struct TreeNode{
    int sline;/* 所在行数 */
    int stype;/* 0表示语法单元,1表示一般词法单元,2表示ID,3表示type,4表示int,5表示float */
    char* name;
    char* sID;
    int snum;
    double snum1;
    struct TreeNode* child;/*最左边的孩子节点 */
    struct TreeNode* silbing;/*右边紧挨着的兄弟节点 */
}Node;
```

对树的相关操作:

在增加孩子节点时，由于 bison 用的是自底向上的语法分析，因此也更新该语法单元的行数。

```
/*增加该节点的孩子节点 */
void CTchild(Node*node,Node* child){
    if(node !=NULL && child!=NULL){
        node->child = child;
        node->sline = child->sline;
    }
}
```

```
/*增加该节点的兄弟节点 */
void CTsilbing(Node* node,Node* silbing){
    if(node !=NULL && silbing!=NULL){
        node->silbing = silbing;
    }
}
```

此外，由于词法错误会导致语法产生式匹配错误，为了不多报错并且能在词法错误的情况下不打印语法树，报错函数如下：

```
void yyerror(char* mag){
  if(flexflag==0){
  syntaxflag=1;
  printf("Error type B at line %d: %s.\n",line,mag);}
  else {flexflag=0;syntaxflag=1;}
}
```