南京大学本科生实验报告

课程名称：**编译原理**

学院	计算机科学与技术	专业（方向）	计算机科学与技术
学号	**201220215**	姓名	**田馥雯**
Email	1259067849@qq.com	实验名称	**实验一：词法分析和语法分析**

实验内容

已实现全部必做及选做内容

- 查出C--源代码中可能包含的下述几类错误：
  1. 词法错误；
  2. 语法错误：识别八进制数和十六进制数、识别指数形式的浮点数、识别“//”和“/.../”形式的注释。
- 接收一个输入文件名作为参数，输出相关的词法或语法有误的信息或者按照先序遍历的方式打印构造好的语法树中每一个结点的信息。

通过`Makefile`生成可执行文件

```
make clean
make
./parser test.cmm
```

核心代码

代码结构

- `main.c` 获取文件输入，初始化错误标志，通过词法分析和语法分析检验错误，构建并打印语法树
- `tool.c/tool.h` 构建语法树过程中所用到的功能函数
- `lexical.l` 匹配所有词法单元，建立相应节点并返回`token`
- `syntax.y` 利用产生式匹配并建立语法树

语法树结点结构

```
/* AbstractSyntaxTree 抽象语法树 */
typedef struct AbstractSyntaxTree
{
    int lineNumber; // 行数
    int abstractSyntaxTreeNodeType; // 节点类型
    char *name; // 节点名称
    char *value; // 节点值
    struct AbstractSyntaxTree* firstChildrenNode; // 第一个孩子节点
    struct AbstractSyntaxTree* nextSiblingNode; // 下一个兄弟节点
}AST;
```

功能函数

```
/* createNewAbstractSyntaxTreeNode 创建新语法树节点 */
AST *createNewAbstractSyntaxTreeNode(char *name, char *value, int type, int
lineNumber);
```

```
/* addAbstractSyntaxTreeNode 添加语法树节点 */
void addAbstractSyntaxTreeNode(AST *pre, AST *next);
```

```
/* deleteAbstractSyntaxTree 删除语法树 */
void deleteAbstractSyntaxTree(AST *root);
```

```
/* printAbstractSyntaxTree 打印抽象语法树 */
void printAbstractSyntaxTree(AST *root, int depth);
```

```
/* printSpaces 打印空格 */
void printSpaces(int count);
```

```
/* stringComparison 字符串比较 */
bool stringComparison(char *src, char *dst);
```

```
/* printNodeMessage 打印信息 */
void printNodeMessage(char *name, char *value);
```

```
/* to_Oct 转换成八进制数 */
int to_Oct(char *str);
```

```
/* to_Hex 转换成十六进制数 */
int to_Hex(char *str);
```

```
/* to_Dec 转换成十进制数 */
int to_Dec(char *str);
```

```
/* My_atoi 自定义atoi */
int My_atoi(char *str);
```

总结与感想

代码错误及解决方法

内存溢出

初始结点中`name`和`value`利用静态字符数组，当语法树结点过多时会导致内存溢出

```
    char name[10000]; // 节点名称
    char value[10000]; // 节点值
```

修改为建立过程中动态分配

```
    newAbstractSyntaxTreeNode->name = malloc((strlen(name) + 1) * sizeof(char));
    newAbstractSyntaxTreeNode->value = malloc((strlen(value) + 1) *
sizeof(char));
```

为空时也可通过

语法分析匹配时加入

```
| error{
    errorSyntaxFlag++;
}
```

感悟

还是学到了很多的，就是有点拖deadline，下一个实验一定早点开始